CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75789 ARIEL GUSTAVO RODRIGUES DA S
24PRG204 - 1 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=2, b=7, lim=2800,       resposta=2238600
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=8041, Ls=9527,         Quantidade=168
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1487012,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 3 ou 7 abaixo de 3000
2. Descubra quantos primos existem entre 3180 e 6396 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 751607

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75796 CASSIANO MAGNO CHAGAS E SA
24PRG204 - 2 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=5, b=7, lim=2300,        resposta=830467
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=8035, Ls=8731,        Quantidade=78
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=331235,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 19 ou 2 abaixo de 3400
2. Descubra quantos primos existem entre 3022 e 3947 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 846295

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75808 CHRISTOPHER BUCH FILIPAK
24PRG204 - 3 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=13, b=5, lim=4700,       resposta=2885263
```

Desafio Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=4992, Ls=5791,        Quantidade=93
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1028911,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 5 ou 11 abaixo de 2100
2. Descubra quantos primos existem entre 7876 e 7970 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 125188

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75815 ERAN MARTINEZ RAMOS 24PRG204 - 4
/ _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=17, b=19, lim=4400,       resposta=1047718
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=5839, Ls=9388,         Quantidade=395
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1463377,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 5 ou 3 abaixo de 4900
2. Descubra quantos primos existem entre 5654 e 6032 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 473389

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75822 HENRIQUE IVANKIO GUSSE 24PRG204 - 5 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=13, b=17, lim=3300,      resposta=716053
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=8312, Ls=8508,        Quantidade=17
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=300838,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 5 ou 19 abaixo de 3000
2. Descubra quantos primos existem entre 5088 e 7456 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 607584

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75839 IAN DOLABELLA OHATA 24PRG204 - 6
/ _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=5, b=11, lim=3000,       resposta=1225233
```

Desafio Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos $hash$, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=6969, Ls=7809,        Quantidade=92
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=637854,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 19 ou 17 abaixo de 3100
2. Descubra quantos primos existem entre 3734 e 9290 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 683456

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75846 JOAO BONFIN LINO 24PRG204 - 7
/ _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=17, b=19, lim=4500,       resposta=1096621
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=4966, Ls=6674,         Quantidade=197
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1663166,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 2 ou 13 abaixo de 4700
2. Descubra quantos primos existem entre 4084 e 9040 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 749887

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75853 JOAO LUCA FERNANDES DE OLIVE
24PRG204 - 8 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=13, b=5, lim=4100,       resposta=2194920
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,          Quantidade=689
Li=6844, Ls=8013,          Quantidade=128
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=510996,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 3 ou 2 abaixo de 4500
2. Descubra quantos primos existem entre 8338 e 9707 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 378995

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75860 JOAO PEDRO DE BRITO DUARTE
24PRG204 - 9 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=7, b=5, lim=3400,       resposta=1812930
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=7324, Ls=9528,         Quantidade=246
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=951063,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 19 ou 2 abaixo de 2300
2. Descubra quantos primos existem entre 5585 e 6012 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 674005

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75877 JOAO PEDRO MARIANO SUEKE
24PRG204 - 10 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=7, b=19, lim=4600,       resposta=1992593
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que *limite* somado com a soma dos divisíveis por b (menores que *limite*). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=4836, Ls=6818,        Quantidade=226
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=640506,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 2 ou 11 abaixo de 4000
2. Descubra quantos primos existem entre 4142 e 9499 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 538040

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75884 JOAO VICTOR GOMES VILELA GER
24PRG204 - 11 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=17, b=7, lim=4400,       resposta=1871275
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=6745, Ls=8416,        Quantidade=182
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=708839,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 5 ou 13 abaixo de 4900
2. Descubra quantos primos existem entre 5447 e 8530 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 995173

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75989 LUCAS ANTOSZCZYSZEN 24PRG204 -
12 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=2, b=5, lim=2300,       resposta=1585850
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=7413, Ls=8388,         Quantidade=110
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=364434,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 11 ou 13 abaixo de 3600
2. Descubra quantos primos existem entre 5494 e 9607 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 766270

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75891 LUIZ FELIPE GUEDES BUCCHERI
24PRG204 - 13 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=2, b=5, lim=3200,       resposta=3070400
```

Desafio Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=3195, Ls=9258,        Quantidade=695
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,      Soma=60696
Limite=1195589,      Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 5 ou 2 abaixo de 4300
2. Descubra quantos primos existem entre 5725 e 6155 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 560845

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75903 MANUELLA LEAL DE MEIRELLES F
24PRG204 - 14 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=3, b=2, lim=3000,       resposta=2998500
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=3028, Ls=3043,        Quantidade=2
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=459854,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 19 ou 5 abaixo de 5000
2. Descubra quantos primos existem entre 3027 e 6634 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 946079

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75910 MARIA EDUARDA DA SILVA 24PRG204 - 15 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=7, b=2, lim=2600,       resposta=1930872
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=3153, Ls=9473,         Quantidade=728
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=453116,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 3 ou 17 abaixo de 2200
2. Descubra quantos primos existem entre 4010 e 6418 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 998081

Responda aqui:

1	2	3

204-75910 -

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75927 MATHEUS AITA FABRICIO DE CAR
24PRG204 - 16 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=13, b=7, lim=3400,       resposta=1205495
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,          Quantidade=689
Li=5966, Ls=8330,          Quantidade=264
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=344705,       Soma=257114
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 11 ou 5 abaixo de 2900
2. Descubra quantos primos existem entre 4777 e 5581 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 233634

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75934 MATHEUS LEANDRO DE BITENCOUR
24PRG204 - 17 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=11, b=13, lim=4200,       resposta=1418514
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=6897, Ls=9615,        Quantidade=300
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=649112,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 11 ou 17 abaixo de 2200
2. Descubra quantos primos existem entre 7602 e 9663 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 246495

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75941 MAYCON TAVARES WOLPE 24PRG204 - 18 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=19, b=7, lim=2600,       resposta=634776
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que *limite* somado com a soma dos divisíveis por b (menores que *limite*). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,          Quantidade=689
Li=6066, Ls=7648,          Quantidade=180
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1699648,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 13 ou 3 abaixo de 4100
2. Descubra quantos primos existem entre 6845 e 8851 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 386468

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75958 PEDRO TOLEDO LEAL 24PRG204 - 19
/ _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=19, b=7, lim=4900,       resposta=2253879
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,          Quantidade=689
Li=7404, Ls=8627,          Quantidade=135
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=998648,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 17 ou 13 abaixo de 2200
2. Descubra quantos primos existem entre 4275 e 7744 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 55157

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75965 PHELIPE GABRIEL LIMA DA SILV
24PRG204 - 20 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=3, b=2, lim=4000,       resposta=5332667
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,         Quantidade=689
Li=3011, Ls=4920,         Quantidade=226
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=1333765,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 19 ou 13 abaixo de 3600
2. Descubra quantos primos existem entre 3160 e 7129 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 419955

Responda aqui:

1	2	3

CEP - UFPR - UP - UTFPR - PUCPr –
03/06/2024 - 13:26:50.8
Prof Dr P Kantek (pkantek@gmail.com)
Programas a desenvolver VIVX072b, V: 1.21
75972 REBECA CABRAL DOS SANTOS
24PRG204 - 21 _____ / _____ / _____

Iniciando na programação

Citando Aristóteles, *Tudo o que precisamos aprender, aprendemos fazendo* (no seu livro *Ética*), vamos concordar que para se desenvolver na habilidade de programar computadores, o que é necessário fazer é programar. Na verdade, nada é muito diferente em muitas outras habilidades humanas: cirurgia, conserto de veículos, alta gastronomia, tocar piano ou qualquer outro instrumento, escrever bem, desenhar maravilhosamente, erguer uma parede, ... a lista é interminável.

Dito isto, a seguir vem uma lista de programas que você deve produzir. A linguagem escolhida pode ser qualquer uma:

C durante muito tempo a linguagem usada em universidades e escolas para aprender

C++ uma evolução de C, sobretudo para simplificar as entradas e saídas de dados

Python a nova queridinha no processo de aprender a programar. É a que garante maior produtividade sobretudo no início do aprendizado

Javascript um bocadinho mais complicada, por ter que usar um browser durante sua execução, mas tem a vantagem de permitir um visual GUI bem elegante e fácil (por conta da dupla HTML/CSS)

APL meio antiga, mas insuperável para construir programas pequenos embora muito poderosos

Java até pouco tempo a preferida dos desenvolvedores, roda em qualquer coisa aparentada ao computador

Outras Assembler, Octave, Maple, Pascal, Fortran, Lisp, Prolog, ... a lista é quase interminável.

Enfim, a lista acima apareceu para subsidiar o comentário de que **não importa a linguagem escolhida** o algoritmo é sempre o mesmo ou muito parecido. E, o algoritmo é sempre o mais difícil de produzir. Então, de fato, a linguagem utilizada é somenos importante.

Em resumo, use o que quiser e dentro da linguagem escolhida use o recurso que quiser. O importante é achar o resultado:

Programa 1: Múltiplos de um número

Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, teremos:

```
3, 5, 6 e 9
```

A soma desses números é 23.

Resolvendo o caso

para uma instância real. Suponhamos $a = 3$ e $b = 5$ e $limite = 1000$. Uma maneira simples de resolver o caso é:

```
limite = 1000 - 1
soma = 0
para i de 1 até limite passo 1
   se (i%3==0) || (i%5==0)
      soma=soma+i
   fim{se}
fim{para}
escreva(soma)
```

Este programa deve imprimir 233.168 que é a soma pedida.

O próximo passo é deixar o programa genérico com as variáveis a, b e $limite$ sendo lidas no início do processamento.

Teste o seu programa com as seguintes triplas

```
a=7, b=11, lim=2600,       resposta=747741
a=13, b=11, lim=2300,      resposta=424435
```

Desafio

Se você for questionado sobre um limite de 1.000.000.000 este programa pode demorar um tanto. Uma estratégia um pouco mais eficiente, pode ser obter a soma dos divisíveis por a menores que $limite$ somado com a soma dos divisíveis por b (menores que $limite$). Só que fazendo isso, você somou os divisíveis por $a \times b$ duas vezes. Vamos olhar o exemplo para $n = 3$ Precisamos somar $3 + 6 + 9 + 12 + ... + 999$ que é igual a $3 \times (1 + 2 + 3 + 4 + ... + 333)$.

Para $n = 5$ precisamos somar $5 + 10 + 15 + ... + 995 = 5 \times (1 + 2 + 3 + ... + 199)$. Note que $199 = 995/5$, mas também $999/5$ em termos de divisão inteira. Se também notarmos que $1 + 2 + 3 + ... + p = \frac{1}{2} \times p \times (p + 1)$, temos:

Se você definir uma função chamada

```
função somds(int n, int lim)
   limite = lim-1
   p = limite / n // (divisao inteira)
   retorne n * (p * (p+1)) / 2
fim{função}
```

O cálculo agora é fácil, bastando fazer:

```
somds(3,1000)+somds(3,1000)-somds(15,1000)
```

Programa 2: Números primos

Um número primo é um número natural maior que 1 que possui apenas dois divisores positivos: 1 e ele mesmo.

Em outras palavras, um número primo não pode ser formado pela multiplicação de outros números naturais menores.

Exemplos de números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

Propriedades importantes dos números primos:

O número 1 não é primo: Ele possui apenas um divisor (ele mesmo). Todos os números pares maiores que 2 são compostos: Eles podem ser formados pela multiplicação de dois números menores (por exemplo, 8 = 2 x 4). A distribuição dos números primos é infinita: Não existe um limite para a quantidade de números primos que existem. Os números primos desempenham um papel fundamental na teoria dos números e em diversas áreas da matemática e da computação: Eles são utilizados na criptografia, na construção de códigos e na decomposição de números em fatores primos. Especificamente na área da computação, eles são a base de geração de pseudo-aleatórios, na construção de tabelas e códigos *hash*, na maioria dos algoritmos de criptografia (RSA, El Gamal, elíptica, etc).

Métodos para encontrar números primos: Básico: contar os divisores do número. Se forem apenas 2 o número (inteiro e maior que 1) é primo. Se forem mais, ele é composto. Crivo de Eratóstenes: Um método antigo e eficiente para encontrar números primos menores que um determinado valor. Teste de Miller-Rabin: Um algoritmo probabilístico para determinar se um número é primo ou composto. Outros métodos: É moda procurar estes algoritmos. Se alguém achar um que seja determinístico e de complexidade menor que exponencial, seu nome vai para a história.

Curiosidades: O maior número primo conhecido: Em dezembro de 2023, o maior número primo conhecido é $2^{(82,589,933)} - 1$, com 25.097.524 dígitos decimais. Os números primos gêmeos: São dois números primos consecutivos, como (3, 5) e (11, 13). Os números primos palíndromos: São números primos que lêem iguais em ambas as direções, como $(2, 5, 7, 11, 131)$.

O algoritmo que você deve implementar é

```
leia (a,b)
ct = 0
var = 2
enquanto (a<=b)
   var=2
   enquanto (var<a)
      se a%var == 0
         saia // limite é composto
      senão
         var = var + 1
      fim{se}
   fim{enquanto}
   se var == limite
      ct=ct+1  // limite é primo
   fim{se}
   a=a+1
fim{enquanto}
escreva(ct)
```

Para testar seu programa, use considere que entre 2 e 100 há 25 primos. Outras opções de teste:

```
Li=2780, Ls=8777,        Quantidade=689
Li=6828, Ls=7472,        Quantidade=67
```

Programa 3: Sequência de Fibonacci

A Sequência de Fibonacci, também conhecida como Sequência de Leonardo, é uma série de números inteiros positivos, começando normalmente por 0 e 1, na qual cada termo subsequente é a soma dos dois anteriores.

Fórmula: O termo n da sequência de Fibonacci pode ser calculado pela fórmula: $F_n = F_{n-1} + F_{n-2}$ Onde: F_n é o termo n da sequência, F_{n-1} é o termo anterior $(n - 1)$ da sequência e F_{n-2} é o termo anterior $(n - 2)$ da sequência.

Os primeiros termos da sequência são: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ O quinto termo da sequência (F_5) é igual a 5, pois $F_5 = F_4 + F_3 = 3 + 2 = 5$. O décimo termo da sequência (F_{10}) é igual a 55, pois $F_{10} = F_9 + F_8 = 34 + 21 = 55$.

A razão entre dois números consecutivos da sequência tende para o Número Áureo, representado pela letra grega ϕ (fi). O Número Áureo é um número irracional aproximadamente igual a 1,6180339887... A sequência de Fibonacci está presente em diversos elementos da natureza, como na disposição das pétalas de algumas flores, nas ramificações de árvores e na estrutura de conchas. A sequência de Fibonacci tem diversas aplicações em matemática, computação, arte e arquitetura.

O programa que você deve construir é para obter a soma dos números da sequência de Fibonacci que são PARES e menores ou igual a um determinado limite dado.

Por exemplo, se o limite for 100, a sequência é 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 e a soma buscada é $2 + 8 + 34 = 44$.

O algoritmo em questão é

```
soma=0
leia(limite)
a=1
b=2
soma=2
c=3
enquanto (c<=limite)
   a=b
   b=c
   c=a+b
   se c%2==0
      soma=soma+c
   fim{se}
fim{enquanto}
escreva(soma)
```

Para testar seu programa use estes casos

```
Limite=121000,       Soma=60696
Limite=704714,       Soma=1089154
```

☞ Para você fazer

1. Encontre a soma de todos os múltiplos de 2 ou 17 abaixo de 2500
2. Descubra quantos primos existem entre 4014 e 7144 (incluidos os extremos).
3. O limite da sua implementação para os números de Fibonacci (soma dos pares) é 126190

Responda aqui:

1	2	3