PRG18

O tipo string como um vetor de caracteres. Em C, o tipo elementar é *char* cujo comprimento é 1 byte. Suas constantes são escritas entre aspas simples, mas podem ser formadas com `\algumacoisa`. Esta barra invertida é chamada ESCAPE, e as principais são `\b`=backspace, `\0`=nulo, `\n`=new line, `\a`=alarme, entre outras.

Em C, é muito ruim manipular (mover, testar, preencher, ...) vetores de char.

> Como em C não há o string, o que existe é o array de char. É um saco, a qualquer momento arrisca-se uma *memory violation*.

```
#include<string.h>
char x[tamanho];
printf("%s\n",x); // para imprimir
strcpy(x,"conteudo para x"); // para assinalar
// é ilegal fazer x="conteudo para x";
strcmp(x,y); //x=destino, y=origem -- é ilegal x<"conteudo"
strcat(x,y); // para concatenar
strlen(x); // para saber o tamanho
```

Daí que em C++ surgiu um novo tipo primitivo chamado *string*. (Eventualmente pode ser necessário `#include<string>`, mas aparentemente não é). Veja um exemplo

```
string aa = "Ola mundo !";
string x("ola mundo");  // inicialização alternativa...
string y={"vamos ver"}; // idem...
```

Note as aspas duplas, para diferenciar do tipo char. (Em Python não é assim...)

Não aparece, mas TODO string termina por um caracter `\0`. É assim que o C (e o C++) sabem onde um string termina.

Registre-se que tecnicamente um string é um array de char, mas ele pode ser entendido como um novo tipo sendo na realidade uma classe (que vem a ser um "tipo" mais "inteligente" por exemplo na comparação entre strings ou no seu redimensionamento).

Os operadores deste novo tipo são: = atribuição, + concatenação, [] acessar caracteres individualmente, além de << escrever e >> ler. Maior (>) e menor (<) com o significado de vem antes ou vem depois. Também existem as conversões de tipos. Veja exemplos

```
string a0="Ola"
string a1=" Mundo"
string aa=a0+a1;
aa[4]='Z';  // lembre que começa em zero...
cout<<aa;  // mostra ola Zundo
```

O tamanho de uma string é dado por

```
string x = "Ola mundo";
cout<<x.length();  // ou
cout<<x.size(); // ambos dão 9
string z;
z=x; // funciona perfeitamente
```

Na leitura depois de um `string z;` fazer `cin<<z;` tem um problema. A leitura é interrompida após o primeiro espaço em branco, tabulação ou < *enter* >. Para obter a digitação completa até o < *enter* >, supostamente o final da entrada, deve-se mudar o comando de entrada para

```
string z;
getline(cin,z);
```

Já o cout não precisa mudar e funciona igual ao que já se sabe.

Na comparação, dois strings podem ser iguais (mesmo tamanho e mesmo conteúdo). Veja os exemplos

```
string x,y,z;
x="ola mundo";
y="ola";
z="OLA MUNDO";
cout<<(x>y);  // dá 1
cout<<(z<x);  // dá 1
if (msg1==msg2){cout<<"iguais";} else {cout<<"difs";}
```

O significado de > e < em strings é "vem depois (>), e vem antes (<) na sequência de códigos (ASCII, EBCDIC, UNICODE, ...) empregado. Este tipo de comparação é fundamental para por textos em ordem alfabética.

Arquivos

Este é um capítulo extenso da programação. Aqui, só alguns pitacos

```
#include<stream>  // arquivos com conteúdos string...
...
    ofstream arqs  // objeto arquivo de saída
    ifstream arqe // idem para arquivo entrada
    arqs.open("/p/ufpr/lixo1.dad");
    arqs<<"vamos ver se vai"<<endl;
    arqs.close();
```

Daí lá no disco...

```
C:\p\ufpr>type lixo1.dad
vamos ver se vai
```

Para fazer o contrário (ler dados do disco...)

```
char data[100];
arqe.open("...nome do arquivo...");
arqe>>data;
...
arqe.close()
```

Para encerrar, um ciclo completo de ler e gravar em arquivos:

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
   string line;
   ifstream meuarq("/p/ufpr/lixo1.dad");
   if (meuarq.is_open()){  // deu certo abrir ?
      while (!meuarq.eof()){
         getline(meuarq,line);
         cout<<line<<endl;
      }
      meuarq.close();
   }
   else {cout<<"deu xabu no arquivo"<<endl;}
   return 0; }
```

(Palíndromo)

Escreva um programa C++ que leia uma frase e informe se ela é (1) ou não é (0) um palíndromo. Exemplos de palíndromos

```
socorram me subi no onibus em marrocos
a droga da gorda
roma me tem amor
oto come mocoto
sairam o tio e oito marias
me ve se a panela da moca e de aco madalena pela vem
a diva da vida
luz azul
ato idiota
o treco certo
a base do teto desaba
atila toledo mata modelo talita
anotaram a data da maratona
o romano acata amores a damas amadas e roma ataca o namoro
sa da tapas e sapatadas
seco de raiva coloco no colo caviar e doces
assim a aia ia a missa
o cid e medico
```

```
erro comum ocorre
a cera causa sua careca
morram apos a sopa marrom
solo di sol a los idolos (espanhol)
esope reste ici et se repose (francês)
Elle va à la vache (idem)
```

```
#include<iostream>
#include<string>
using namespace std;
int main(){
   string frase;
   int pali,ini,fin;
// cin>>frase; pega só até o primeiro branco
   getline(cin, frase);
   pali=1;
   fin=frase.length()-1;
   ini=0;
   while(fin>=ini){
      while(frase[ini]==' '){ini++;}
      while(frase[fin]==' '){fin--;}
      if (frase[ini]!=frase[fin]){pali=0;}
      fin--;
      ini++;
   }
   cout<<pali;
}
```

(USP6.6: frase e palavra)

COMP89 - Dados dois strings (um contendo uma frase e outro contendo uma palavra) determine o número de vezes que a palavra ocorre na frase. Por exemplo, para a frase "ANA E MARIANA GOSTAM DE BANANA"tem-se que a palavra "ANA"ocorre 4 vezes na frase.

```
#include<iostream>
#include<math.h>
#include<string>
using namespace std;
int main(){
  string fr,pa;
  int i,j,nao,qtd=0;
  getline(cin,fr);
  getline(cin,pa);
  // tamanho=fr.length()-1;
  qtd=0;
  for (i=0;i<fr.length()-pa.length()+1;i++){
      nao=0;
      for (j=i;j<pa.length()+i;j++){
        if (fr[j]!=pa[j-i]){nao=1;}
      }
      if (nao==0){qtd++;}
  }
  cout<<qtd;
}
```

Escreva um programa C++ que leia uma frase de até 80 caracteres e imprima cada palavra em nova linha. O separador de palavras é um e só espaço em branco.

```
#include<iostream>
#include<string>
using namespace std;
int main(){
  string fra="ivo viu a uva e a banana";
  string pal,pal1;
 // getline(cin,fra);
  int i,j,k,tam;
  tam=fra.length();
  i=0;
  while (i<tam){
      j=i;
      while (fra[j]!=' '){
```

```
        j++;
      }
      cout<<fra.substr(i,j-i)<<"-"; // in,tam
      i=j+1;
  }
  cout<<"\b ";
}
```

Conversões

```
#include<iostream>
#include<string>
using namespace std;
int main(){
  string x,y,fra="12340";
  int n;
  float f;
  n=stoi(fra);   // - string to integer
  f=stof(fra);   // - string to float
  cout<<n<<endl;  // 12340
  cout<<f<<endl;  // 12340
  x=to_string(n); // o que for -- para string
  y=to_string(f); // idem -- idem
  cout<<x<<endl; // 12340
  cout<<y<<endl; } // 12340.000000
```

Lembram do palíndromo numérico ?

```
int main(){
  int n,a1,a2,enaoe,j;
  cin>>n;
  enaoe=1;
  j=floor(log10(n));
  while (n>0){
    a1=n%10;
    a2=n/pow(10,j);
    if (a1!=a2){enaoe=0;}
    n=n-a2*pow(10,j);
    n=n/10;
    j=j-2;
  }
  cout<<enaoe; }
```

Agora um outro jeito:

```
int main(){
   int n,i,j,simnao=1;
   string s;
   cin>>n;
   s=to_string(n);
   i=0;
   j=s.size()-1;
   while (i<=j){
      if (s[i]!=s[j]){simnao=0;}
      i++;
      j--;
   }
   cout<<simnao; }
```