Estudando programação

Como já dizia Aristóteles (no livro Ética) *Tudo o que precisamos aprender, aprendemos fazendo....* Na programação não é diferente. Mas, aqui temos uma grande vantagem: um corretor justo, incansável, gratuito, paciente e sempre disponível. Trata-se do computador. Você pode produzir código e o computador vai lhe dizer se o seu código está certo ou não.

Listei abaixo 3 opções onde você pode estudar programação. A mais amigável é a OBI, mas nos últimos dias o site estava fora do ar. Deve voltar em breve, mas as outras duas; UVA e Euler também são lugares legais.

1 OBI

OBI é a Olimpíada Brasileira de Informática. É uma iniciativa sensacional para ajudar estudantes brasileiros a se desenvolverem na programação. Qualquer um pode se beneficiar do corretor automatizado de programas. Basta entrar no site da OBI (https://olimpiada.ic.unicamp.br/pratique/) e escolher uma definição à direita na tela. Há dezenas de definições bastante bem escritas e com exemplos de entradas e saídas – o que é uma excelente maneira de entender o que o programa deve fazer. Entendida a encomenda, você deve escrever seu programa (no caso particular de Python, nem programa precisa ser, basta que seja um script funcional. Em C++ basta mandar ver). Uma vez salvo em algum lugar conhecido por você (diretório e subdiretório do seu disco) e com um nome único, basta escolher na tela a linguagem que você usou e o programa que escreveu. Ao clicar em SUBMETER o seu programa será lido, compilado e executado um determinado número de vezes, com outros dados de entrada diferentes daqueles que estão na definição.

Após alguns segundos, a aplicação WEB retorna uma resposta que pode ser

```
Resultado da submissão
TAREFA: Álbum da copa
LINGUAGEM: Python3
Compilação correta
Fase de testes -- Tempo Limite para cada execução: 500 ms
Teste 1: ..... (20 pontos)
Teste 2: ..... (20 pontos)
Teste 3: ..... (20 pontos)
Teste 4: ..... (20 pontos)
Teste 5: ..... (20 pontos)
Total: 100 pontos (de 100 possíveis)
Legenda:
.: resultado correto              X: resultado incorreto
E: erro em tempo de execução      M: Referência a memória inválida
S: programa não gerou saída       T: tempo limite excedido
V: violação de recursos
```

2 UVA

Instalado na Universidad de Valladolid, Espanha (daí o nome UVA) trata-se de um engenho de correção de solução de problemas algoritmicos funcionando no modelo das maratonas. O site é `https://uva.onlinejudge.org/`.

O site foi criado em 1995 por Miguel Angel Revilla, um professor de algoritmos da U. Valladolid. O site entrou em ação para o público em geral em Abril de 1997. Em 2007 o sistema foi migrado (software e hardware) para melhores condições.

O interessado pega uma definição de problema de programação, (há mais de 5000 no site), escreve um programa (em C, Java, Python ou C++) que o resolve e submete ao site o programa fonte que escreveu.

Para auxiliá-lo nesta tarefa o site UVA entrega, para cada programa da lista, um conjunto preparado de dados de entrada e associados a cada um, o conjunto de dados de saída que o programa supostamente deveria produzir. Com essas 3 coisas: definição, dados de entrada e dados de saída esperados, o candidato a programador tem tudo o que precisa para escrever e enviar ao site o programa pedido.

O site, ao receber tal programa, compila-o, e o executa com uma instância nova (inédita) de dados.

Os resultados que o site entrega alguns segundos após a submissão são:

OK o programa que você submeteu foi rodado com dados inéditos e produziu exatamente o resultado esperado.

Limite tempo O programa não terminou no intervalo de tempo a ele alocado. Ou o programa está em loop devido a algum erro nas estruturas de repetição, ou o que é menos comum, é necessário um algoritmo mais eficiente para atender a este problema.

Erro de compilação O nome diz tudo, o site não conseguiu compilar o programa que você submeteu. Ou ocorreu um equívoco na linguagem selecionada ou a linguagem instalada no seu computador não é compatível com aquela usada pelo site. Ou ainda, o que é mais frequente, você cometeu um erro de escrita no seu código.

Resultado diferente O seu programa funcionou, mas produziu resultados distintos dos que o site esperava para aquele problema. Em outras palavras, sua solução é incorreta. A chave para resolver este tipo de erro é uma leitura cuidadosa da definição, sobretudo nos detalhes (chamadas condições de contorno).

Muito importante: na última instrução executada pelo programa em C ou C++, deve-se retornar um 0. Sem isso, o programa acusará erro.

3 Projeto Euler

Este é outro esquema muito interessante para aprender a programar e se desenvolver na nobre arte. Ao contrário da UVA, no projeto Euler não se envia um programa e sim um resultado numérico que geralmente é difícil de se obter. O site é `http://projecteuler.net`.

Aqui, há razões de ordem prática para se buscar algoritmos e programas eficientes. Alguns pedidos se programados sem cuidado podem demorar meses ou anos em um computador comum. Ninguém precisa esperar tanto e segundo os criadores do site, nenhum problema demanda mais do que 1 segundo de CPU.

A questão é que para muitos problemas, tal limite de tempo impõe severas restrições e especialização no algoritmo procurado.

Um caso real que aconteceu comigo. Ao programar um problema (o 104 na lista do Euler) que diz

A sequencia de Fibonacci é definida pela relaç↑o de recorrência $F_n = F_{n-1} + F_{n-2}$ *onde* $F_1 = 1$ *e* $F_2 = 1$. *Deve-se notar que* F_{541} *que contém 113 dígitos é o primeiro número de Fibonacci para o qual os últimos 9 dígitos s↑o pandigitais 1-9 (ou seja contém todos os dígitos de 1 a 9 n↑o necessariamente em ordem). E,* F_{2749} *que contém 575 dígitos é o primeiro número de Fibonacci para o qual os primeiros 9 dígitos são pandigitais 1-9. Dado* F_k *que é o primeiro número de Fibonacci para o qual os primeiros 9 dígitos E os últimos 9 dígitos são pandigitais 1-9, ache* k.

Minha primeira opção demorou 5 dias rodando para achar o resultado (certo). Essa demora se justificava pois lida-se com números de Fibonacci com mais de 5000 dígitos cada um. Daí, quebrei a cabeça para otimizar o programa e alguns dias depois cheguei a uma versão equivalente que demorou menos de 1 segundo de tempo de relógio.

Seja como for, como você só vai fornecer um resultado, não importa quanto vai demorar, nem qual linguagem/compilador ou ambiente operacional você vai usar. Em tese, poderia até usar lápis e papel apenas. Há usuários cadastrados usando mais de 100 linguagens diferentes na obtenção das soluções.

O site não dá nenhuma dica ou auxílio até você resolver certo o problema. Nessa hora você ganha acesso a uma lista de discussão de todas as pessoas que já resolveram o mesmo problema. Muita da otimização posterior vem das idéias postadas nessa lista.

Alguns problemas são muito difíceis. Pode-se dizer que são adequados apenas para poucas centenas de habitantes deste planeta. A coisa é, como se dizia quando eu era pequeno, *às veras*.

Para treinar, entre no site Euler e cadastre-se fornecendo nome e senha. Agora você tem uma conta no servidor e pode começar a submeter problemas. Escolha o problema 1, cuja definição é *Se listarmos todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, obteremos 3, 5, 6 e 9. A soma desses múltiplos é 23. Ache a soma de todos os múltiplos de 3 ou 5 abaixo de 1000.*

Tente escrever um programa de computador que ache a resposta.

Se não conseguir, pode oferecer a resposta certa que é 233168.

Pronto, você já é membro da coletividade mundial do projeto Euler.

