Décimas Primeiras Olimpíadas Nacionais de Informática

1º Problema
Ladrilhando salas quadrangulares

Alvarinho Chiça é um conhecido arquitecto, que gosta muito de projectar salas com formas estranhas. Recentemente, tem-se dedicado a salas quadrangulares, que até nem são muito complicadas, excepto quando os lados não são perpendiculares uns aos outros. Neste caso fica um bocado aborrecido colocar os ladrilhos no chão, e há muito desperdício, com todos os ladrilhos que é preciso partir para ajustar o chão às paredes.

O que se pretende neste problema é escrever um programa para ajudar Alvarinho Chiça a calcular o número de ladrilhos rectangulares necessários para cobrir o chão de uma sala quadrangular. Considere que a planta da sala está desenhada num referencial cartesiano, com a parede "principal" sobre o eixo positivo dos X e o canto "principal" na origem. Por outras palavras, as extremidades da parede principal estão nos pontos de coordenadas (0, 0) e (a, 0), com a > 0. Os outros dois vértices do quadrilátero estão nos pontos de coordenadas (b, c) e (d, e), sabendo-se que c >0, e>0, e que estes dois outros vértices não são coincidentes. Ou seja, temos um quadrilátero definido pelos pontos (0, 0), (a, 0), (b, c) e (d, e), por esta ordem, totalmente incluído no semi–plano y >= 0. Todas as coordenadas são números inteiros, e o são tais que o quadrilátero obtido é garantidamente convexo.

Os ladrilhos são rectangulares e todos iguais, e começam a colocados a partir da origem com a base contra a parede principal, sendo cortados sempre que necessário. A parte não utilizada de um ladrilho cortado é deitada fora (mesmo que pudesse ser reutilizada noutra parte da sala, mas enfim…) Os ladrilhos da segunda fila colocam-se sobre os da primeira fila, com as bases justapostas, e assim por diante, até toda a sala estar coberta. A base de cada ladrilho mede f unidades e a altura g unidades (f e g são números inteiros).

Escreva um programa para calcular o número de ladrilhos rectangulares são necessários para cobrir o chão de uma sala com as características descritas. Os dados para o programa estão num ficheiro de nome INPUT.TXT, com duas linhas: a primeira contém os valores dos números a, b, c, d, e e, e a segunda os dos números f e g. O resultado (o número de ladrilhos) deve ser escrito sem mais nada na única linha de um ficheiro de saída, de nome OUTPT.TXT.

Nota: o seu programa não deve usar nem o terminal, nem outros ficheiros, para além dos dois mencionados.

2º Problema
Números biprimos (?)

Um número primo é um número natural maior que 1 que só é divisível por ele próprio e por 1. A sequência dos números primos é bem conhecida: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, …
Se escrevêssemos isto na base dois teríamos a lista equivalente, mas menos legível: 10, 11, 101, 111, 1011, 1101, 10001, 10011, 10111, 11101, 11111, 100101, …

Pois bem, considera a seguinte definição de número *biprimo*: um número biprimo é um número primo cuja representação na base dois tem um número primo de dígitos '1'.

A sua tarefa neste problema é escrever um programa para listar num ficheiro de texto de nome OUTPUT.TXT, um por linha, todos os números biprimos existentes num certo intervalo, por ordem crescente de *binariedade*. Diz-se que um número é mais binário que outro se a sua representação na base dois tiver mais dígitos '1' que a do outro, ou, tendo o mesmo número de dígitos '1', se for maior que o outro. Se não houver números biprimos no intervalo, o programa escreve na primeira linha do ficheiro de saída a mensagem "não há".

Os limites do intervalo a considerar vêm na primeira linha de um ficheiro de texto de nome INPUT.TXT, separados por um espaço (primeiro o limite inferior, depois o limite superior). O programa começa por

pedir os limites do intervalo, e depois lista os números, por ordem crescente, ou então afixa a mensagem "não há", se não houver números biprimos no intervalo indicado. Considere que os limites fazem parte do intervalo.

Nota: o seu programa não deve usar nem o terminal, nem outros ficheiros, para além dos dois mencionados.

Requisitos técnicos:

1. Os programas devem ser escritos em Pascal ou C/C++, e correr numa janela de DOS.
2. Os programas devem realizar a tarefa pedida, e nada mais, e não devem usar outros recursos para além dos estritamente necessários para isso.
3. Os programas-fonte devem ter o nome LADR, para o primeiro problema e BIPR, para o segundo, com a extensão da linguagem usada; os executáveis chamam-se LADR.EXE e BIPR.EXE.
4. Os programas Pascal devem estar num único ficheiro; os programas C/C++ podem ser organizados em vários módulos, e o requisito do número anterior aplica-se ao ficheiro que contém a função main.
5. Cada ficheiro-fonte deve começar com um comentário com o nome do autor, e outro indicando qual o ambiente para o qual o programa foi desenvolvido.
6. Deve ser fornecido para cada problema, um ficheiro de texto, LADR.TXT e BIPR.TXT, onde o autor descreverá brevemente a estratégia usada e as principais dificuldades.
7. Deve ainda ser fornecido um ficheiro de nome MESSAGE.TXT com as coordenadas do autor (nome, escola, telefone, email, etc.), e que o autor usará para dizer ao júri o que lhe apetecer.
8. Ambos os trabalhos devem ser entregues na mesma disquete, a qual não deve conter nada mais além dos ficheiros identificados nestes requisitos técnicos.

Recomendações

1. Estes problemas devem ser resolvidos individualmente. No entanto, antes de os resolver, pode discuti-los com os seus colegas e com os seus professores. A programação propriamente dita, essa sim, deve ser exclusivamente sua.
2. Se precisar de algum esclarecimento sobre os enunciados, ou sobre outros aspectos desta prova, peça-os por email ao Prof. Pedro Guerreiro, presidente da comissão técnica, para pg@di.fct.unl.pt; as respostas virão por "reply".
3. Entregue o seu trabalho numa disquete trancada, etiquetada com todos os elementos de identificação úteis; não entregue mais nada (além da carta que acompanha a disquete).
4. Verifique cuidadosamente que a sua disquete não tem vírus: se tiver, será destruída e a prova desclassificada.
5. O júri poderá querer recompilar o seu programa; se a compilação falhar, a prova será desclassificada.
6. O júri apreciará o texto do seu programa: seja cuidadoso, claro, asseado, económico e use o seu melhor estilo de programação.
7. Se o seu programa acabar por ser muito parecido com os dos seus colegas que também concorrem, diga isso claramente, (no ficheiro message.txt), explicando porquê; caso contrário a sua prova pode ser desclassificada.
8. Respeite escrupulosamente os requisitos técnicos; se não o fizer, a sua prova pode ser desclassificada.