DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Programação de Computadores (2013-2014)

Oitava folha de Problemas

Matéria abordada: operações lógicas bit a bit; funções (passagem de parâmetros por valor e referência; passagem de tabelas), recursão (funções recorrentes), ciclos (while, do while, for).
Bibliografia: Capítulo 5, 6 e 8 de [Oua03].
Da folha 6 em diante estão proibidas as variáveis globais!

1. Neste exercício vai ser solicitado que utilize os operadores lógicos orientados ao bit (o bit i é o bit de peso 2^i). Considere que foi definida uma variável `unsigned char flags`.

   (a) Escreva uma porção de código que, utilizando o operador &, que lhe permite verificar se o bit 2 de `flags` está activo (ou seja igual a 1).

   (b) Escreva uma porção de código que, utilizando o operador `|`, que lhe permite colocar o bit 4 de `flags` activo (ou seja igual a 1).

   (c) Escreva uma porção de código que, utilizando o operador $\wedge$ que lhe permite inverter o valor do bit 1 de `flags` (ou seja se estava 0 passa a estar a 1 e vice-versa).

   (d) Fazendo `flags = 0x07` diga qual seria o resultado das seguintes instruções (consideradas de forma independente):

   i. `flags <<= 2;`
   ii. `flags >>= 1;`

2. O factorial de n, $n!$, é dado por $\prod_{i=1}^{n} i$.

   (a) Escreva o protótipo (cabeçalho ou assinatura) de uma função que dado um inteiro n ($n \geq 0$) calcula $n!$.

   Se o número dado for negativo a função devolve 0 (o que equivale a um valor incorrecto).

   Nos restantes casos devolve o valor de $n!$. Recorde que 0! vale 1.

   (b) Implemente essa função de forma **recorrente**[1].

   Note que:

   - Caso base: Se n for igual a zero o factorial vale 1;
   - Caso recorrente: Se n maior que zero, $n! = n \cdot (n-1)!$

   (c) Escreva um programa principal que lhe permita testar a função. Utilize valores pequenos para n, $n \leq 12$ (pois num sistema de 32 bits `INT_MAX = 2 147 483 647`, 12! = 479 001 600 e 13! = 6 277 020 800).

3. (a) Implemente a função de protótipo `bool primo(int n);` que devolve `true` se `n` for primo e `false` caso contrário.

   Um número inteiro estritamente positivo é primo se apenas for divisível por si próprio e por um. Note que o número 1 não é primo!

   (b) Escreva um programa principal que pede um inteiro estritamente positivo ao utilizador e indica se esse número é primo ou não, usando a função anterior.

[1] Também usualmente designada por recursiva.

(c) Complete o programa principal da alínea anterior de forma a mostrar todos os números primos menores ou iguais ao número dado pelo utilizador. Utilize a função implementada na alínea 3a.

4. Copie a função implementada na alínea 3a para o ficheiro `f7e4.cpp`.

   (a) Escreva uma função que dado um inteiro estritamente positivo, uma tabela de números inteiros e a dimensão da tabela, preenche essa tabela com todos os número primos menores ou iguais ao número dado (ou até esgotar a capacidade da tabela). A função retorna o número de elementos armazenados na tabela.

   Implemente essa função utilizando a função implementada na alínea 3a (e reciclando o programa escrito na alínea 3c).

   No índice 0 da tabela deverá estar o valor 2, se o número dado for superior a 1.

   (b) Escreva um programa principal que lhe permita testar essa função, e que apresente os valores que foram armazenados na tabela pela função da alínea anterior.

5. Copie a função implementada na alínea 3a para o ficheiro `f7e5.cpp`.

   (a) Escreva uma função que dada uma tabela de números inteiros (e a dimensão da tabela) devolve o número de números primos contidos nessa tabela. Implemente a função utilizando a função implementada na alínea 3a.

   (b) Escreva programa principal que lhe permita testar essa função (declare e inicialize uma tabela que contenha entre os seus elementos alguns números primos).

   Recorda-se que 2, 3, 5, 7, 11 , 13, 17, 19, 23, 29, 31, 37, 41, 43 são primos.

6. (a) Escreva o protótipo duma função, `mediaTab` que calcula a média dos elementos de uma tabela, que são números reais. Considere que o número de elementos na tabela é também um parâmetro de entrada. A média de uma sequência de números é a sua soma dividida pelo número de elementos.

   Implemente essa função.

   (b) Implemente uma função que pede ao utilizador o número de elementos que deseja introduzir numa tabela (compreendido entre 0 e `maxtab`) e seguidamente lê esse número de elementos e armazena-o numa tabela.

   Considere que o prototipo da função é:

   ```
   void leTabDouble(double tab[], int maxtab, int &util);
   ```

   em que `maxtab` é a dimensão máxima da tabela `tab` na qual devem carregados os valores lidos. A função `leTabDouble` devolve através de `util` o número de elementos efectivamente armazenados em `tab`, o qual deve ser sempre menor ou igual a `maxtab`.

   (c) Refaça a alínea anterior, mas agora considere o seguinte protótipo para a função `leTabDouble`:

   ```
   int leTabDouble(double tab[], int maxtab);
   ```

   em que `maxtab` é a dimensão máxima da tabela `tab` na qual devem carregados os valores lidos. A função `leTabDouble` retorna o o número de elementos efectivamente armazenados em `tab`, o qual deve ser sempre menor ou igual a `maxtab`.

   (d) Escreva um programa que lhe permita testar as duas funções criadas. Deve usar `leTabDouble` para carregar valores na tabela e `mediaTab` para calcular a média dos elementos da tabela lida.

   Defina uma constante `const int DIM = 8`, para definir a capacidade da tabela utilizada.

   Teste o seu programa tomando nota dos valores introduzidos e da média esperada comparando-a com o resultado obtido.

7. Na disciplina de Álgebra Linear estuda cálculo matricial. Assuma que desejamos implementar operações sobre matrizes bidimensionais de dimensão $n \times m$ e $m \times k$, em que o número máximo de linhas e de colunas é dado pela constante DIM_MAX, de valor igual a 5 (logo $n, m, k \leq 5$).

   Escreva as seguintes funções:

   (a) adição de duas matrizes, usando a função com o seguinte protótipo:

   ```
   void adicionaMatrizes(const float a[][DIM_MAX], const float b[][DIM_MAX],
                         float soma[][DIM_MAX], int nlin, int ncol);
   ```

   sendo `soma` a matriz soma (todas as matrizes são de dimensão útil `nlin`×`ncol`).

   (b) multiplicação de duas matrizes:

   ```
   void multiplicaMatrizes(const float a[][DIM_MAX], const float b[][DIM_MAX],
                           float prod[][DIM_MAX],
                           int nlin_a, int ncol_a, int ncol_b);
   ```

   em que a dimensão útil da matriz `a` é `nlin_a`×`ncol_a`, a dimensão útil da matriz `b` é `ncol_a`×`ncol_b` e a matriz `prod` terá dimensão útil `nlin_a`×`ncol_b`.

   (c) Implemente um programa principal que lhe permita testar as funções implementadas.

8. (a) Escreva uma função que dado um inteiro devolve $\lfloor\sqrt{n}\rfloor$, ou seja o maior inteiro que não é superior a $\sqrt{n}$. Exemplo: $\lfloor\sqrt{10}\rfloor = \lfloor 3.1623\rfloor = 3$.
   Considere o seguinte protótipo: `int floorSqrt(int n);`
   Para utilizar a função `srqt` precisa de incluir a biblioteca `cmath`, e poderá escolher entre os seguintes protótipos:

   ```
          double sqrt (      double x );
           float sqrt (       float x );
     long double sqrt ( long double x );
   ```

   Nota: Não existindo um protótipo para `sqrt` que tenha um inteiro como argumento deverá fazer um cast explícito para utilizar qualquer uma destas funções

   (b) Para verificar se um número n é primo basta avaliar se é divisível por algum inteiro positivo maior que 1 e menor ou igual a $\lfloor\sqrt{n}\rfloor$.
   Implemente a função de protótipo `bool primo_eficiente(int n);` que devolve `true` se `n` for primo e `false` caso contrário. Na implementação de `primo_eficiente` utilize a função `floorSqrt`, implementada na alínea 8a.

   (c) Considere uma função que dado um inteiro estritamente positivo mostra no ecrã, um por linha, todos os números primos menores ou iguais ao número dado. Proponha um protótipo para essa função e implemente-a utilizando a função implementada na alínea 8b.

   (d) Considere uma função que dada uma tabela de números inteiros (e a dimensão da tabela) devolve o número de números primos contidos nessa tabela. Proponha um protótipo para essa função e implemente-a utilizando a função implementada na alínea 8b.

   (e) Considere uma função que dado um inteiro estritamente positivo, uma tabela de números inteiros e a dimensão da tabela, preenche essa tabela com todos os número primos menores ou iguais ao número dado (ou até esgotar a capacidade da tabela). A função retorna o número de elementos armazenados na tabela. Proponha um protótipo para essa função e implemente-a utilizando a função implementada na alínea 8b.

   (f) Escreva um programa principal que lhe permita testar as funções implementadas.

TPC Crie um programa que peça ao utilizador nome de uma pessoa (string no estilo C), e o ano do seu nascimento (unsigned int) Uma vez recolhida toda a informação apresente-a no ecrã, colocando em cada linha a informação relativa a cada pessoa, assinalando a(s) mais idosa(s). Considere que o número máximo de valores que será possível introduzir está definido numa constante `PESSOAS_MAX` à qual atribuirá um valor (maior do que 0) à sua escolha. Considere que cada nome poderá ter no máximo 56 caracteres úteis – defina a constante `NOME_MAX_SIZE` e atribua-lhe o valor 57. Nesse programa utilizará as funções `le_dados` e `apresenta` que seguidamente se descrevem:

(a) A função, `le_dados`, que solicita ao utilizador:

- quantos dados quer introduzir (maior que 0 e menor ou igual a PESSOAS_MAX);
- dado o valor anterior solicita então o nome e a data de nascimento de de cada pessoa (por esta ordem), os quais armazenará em duas tabelas.
- a função deve retornar o número de elementos introduzidos em cada tabela.

Elementos na mesma posição nas duas tabelas correspondem à mesma pessoa.

Comece por definir o protótipo (ou assinatura) da função.

(b) Escreva uma função, `apresenta`, que apresente no ecrã o nome, o ano de nascimento de cada elemento armazenado nas tabelas (cada par de valores deverá surgir na mesma linha do ecrã). Os nomes dos elementos mais idosos devem ser precedidos de um `*`.

Comece por definir o protótipo (ou assinatura) da função.

Sugestão: Para facilitar a identificação das pessoas mais idosas (nascidos no mesmo ano, o menor) faça uma função auxiliar que lhe devolve o menor ano de nascimento.

Para armazenar os nomes das pessoas (que pode ser formado por mais do que uma palavra) utilize uma variável declarada como se exemplifica:

```
//Cada nome no máximo com 56 caracteres úteis.
char nomes[PESSOAS_MAX][NOME_MAX_SIZE];
```

Use `cin.getline(nomes[i], sizeof(nomes[i]));` em que `i` $\in$ `[0..PESSOAS_MAX-1]`, para garantir que o utilizador nunca pode tentar dar um nome que exceda a capacidade reservada (fixa e igual a `NOME_MAX_SIZE`). Terá de utilizar `cin.ignore(1000,'\n')` em cada ciclo de leitura, mas apenas **após** o `cin` de um número que precede um `getline`.

Referências

[Oua03] S. Oualline. *Practical C++ Programming.* O'Reilly, 3rd edition, 2003.