DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Programação de Computadores (2013-2014)

Nona folha de Problemas

Matéria abordada: funções; operações lógicas bit a bit; estruturas; passagem de parâmetros por valor; passagem de tabelas de estruturas.
Bibliografia: Capítulo 11 e 12 de [Oua03].
Da folha 6 em diante estão proibidas as variáveis globais!

1. (a) Defina uma estrutura (`struct ID`) capaz de armazenar a seguinte informação: nome de uma pessoa, o respectivo número de bilhete de identidade – BI (unsigned int) – e ainda a sua altura em metros (float).

   Considere que o nome é armazenado numa variável do tipo `string` (string no estilo C++).

   (b) Escreva uma função (`le`) que pede ao utilizador a introdução do nome de uma pessoa, do respectivo número de bilhete de identidade – BI (unsigned int) – e ainda a sua altura em metros (float) e seguidamente devolve essa informação numa estrutura.

   Considere o seguinte protótipo: `struct ID le();`

   (c) Escreva uma função (`mostra`) que dada uma `struct ID` apresente no ecrã o nome, o BI e a altura armazenados (o trio de valores deverá surgir na mesma linha do ecrã).

   (d) Escreva um programa principal que utiliza `le` para colocar essa informação numa estrutura e `mostra` para apresentar a informação que foi armazenada.

2. Copie a implementação em `f9e1.cpp` para `f9e2.cpp`. Considere que deseja armazenar um certo número de elemento do tipo `struct ID` (definida na alínea anterior). Considere que o número máximo de elementos está na guardado na constante `MAX_ELEM` elementos e que esta é igual a 100.

   (a) Escreva uma função (`carrega`) que dada uma tabela de `struct ID` e um inteiro, `dim`, pede ao utilizador a introdução da informação relativa a `dim` pessoas (nome, número de bilhete de identidade e altura em metros). Utilize a função `le` que implementou no problema 1b.

   Considere o seguinte protótipo: `void carrega(ID[], int);`.

   Nota: No fim de ler os três elementos de cada estrutura use `cin.ignore()` (para remover do *buffer* o `\n` deixado após a leitura da altura).

   (b) Altere a função `mostra` do problema anterior para mostrar todos os elementos de uma tabela de `struct ID`, dada uma tabela e a sua dimensão. O trio de valores de cada elemento da tabela `struct ID` deverá surgir em cada linha do ecrã.

   (c) Escreva um programa principal que utiliza `carrega` para colocar essa informação numa tabela de estruturas e `mostra` para apresentar a informação que foi armazenada.

3. Considere uma estrutura para armazenar o tempo (ciclo de 24h):

```
struct tempo{
   int hora;     // 0-23
   int minuto;   // 0-59
   int segundo;  // 0-59
};
```

(a) Escreva uma função que dados três inteiros: horas, minutos e segundos, devolve uma `struct tempo`.

(b) Escreva uma função que dada uma `struct tempo` apresenta a respectiva informação numa linha, como se exemplifica:

```
13h45m25s
```

Considere que a função tem o seguinte protótipo:
`void mostraTempo(const struct tempo & dado);`

(c) Escreva uma função que dados dois intervalos de tempo, calcula a sua soma, que fica armazenada no primeiro dos parâmetros. A função deve devolver *false* se não for ultrapassado um período de 24 horas, e *true* caso contrário.

Opte entre os seguintes protótipos:
`bool somaTempo(struct tempo & soma, const struct tempo & parcela);`
`bool somaTempo(struct tempo & soma, struct tempo parcela);` e justifique a sua escolha.

Nota: *Não pode implementar estas duas funções com o mesmo nome* (sobrecarga do nome da função) porque neste caso, não é possível distingui-las no momento da chamada!

Sugestão: Converta os tempos parcelas para segundos, some-os e converta de novo para horas, minutos e segundos; em alternativa pode somar campo a campo e considerar os transbordos sucessivos.

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

4. (a) Implemente uma função que dado um inteiro ímpar, apresente no ecrã um "X" como se ilustra na figura, para o número 5. Se o número dado não for ímpar a função não faz nada.

```
X   X
 X X
  X
 X X
X   X
```

(b) Implemente um programa que peça ao utilizador um inteiro ímpar estritamente positivo compreendido entre 1 e 15 (o programa deverá aceitar apenas valores na gama referida), apresente no ecrã o "X" utilizando a função implementada na alínea anterior.

5. (a) Implemente uma função que dada uma string no estilo C++, de comprimento ímpar apresente no ecrã um "X" como se ilustra na figura para a palavra "FCTUC", Se o comprimento da string não for ímpar a função não faz nada.

```
F   F
 C C
  T
 U U
C   C
```

(b) Implemente um programa que peça ao utilizador uma frase de comprimento entre 1 e 15, terminada por um "." (o programa deverá aceitar apenas frases com o comprimento na gama indicada).

Declare uma variável do tipo `string` e utilize `getline( istream&, string&, char)`. Se a frase tiver comprimento par, acrescente-lhe no final um `'!'`.

Apresente no ecrã o "X" utilizando a função implementada na alínea anterior.

6. (a) Implemente uma função que dada uma letra compreendida entre `'A'` e `'P'`, apresente no ecrã uma pirâmide da forma que se ilustra na figura para o valor F. Se a letra estiver fora desse intervalo, a função não apresenta nada.

```
A
ABA
ABCBA
ABCDCBA
ABCDEDCBA
ABCDEFEDCBA
```

   (b) Implemente um programa que pedindo ao utilizador um carácter de entre A e P, (o programa deverá aceitar apenas valores na gama referida), apresente no ecrã uma pirâmide, utilizando a função implementada na alínea anterior.

7. Considere o programa:

```
//=============================================================
// Pede tres nomes nao vazios e mostra cada um apos cada pedido
// Autor: Nome Apelido               Data:2013/mm/dd
// Objectivo: estudar o alcance e existência de variaveis locais
//            e de variaveis estaticas
//=============================================================
#include <iostream>
using namespace std;

int main()
{
   string nome;      // Local a main()

   // Le tres nomes
   for(int j = 0; j < 3; j++)
   {
      do
      {
         cout << "\nUm nome: ";
         getline(cin, nome);
      }
      while (nome.empty()); // True se nome estiver vazia

      static int passei_aqui = 0;
      int aqui = 0 ;

      ++ passei_aqui;
      ++ aqui;

      cout << "\npassei_aqui " <<  passei_aqui
      << "\n      aqui " <<  aqui << endl;

      nome.clear();  // Torna nome vazia
   }
   return(0);
}
```

   (a) Diga quais as possíveis saídas que espera encontrar. **Responda a esta questão sem transcrever, compilar e executar o programa!**

   (b) Transcreva, compile e execute o programa! Confirme se a sua resposta está correcta.

8. Escreva um único programa que execute as tarefas descritas nos pontos que se seguem.

   (a) Defina um variável (tabela) que lhe permita guardar dez inteiros diferentes e a inicialize-a a seu gosto (inclua pelo menos um número negativo, e um número maior que 255).

   (b) Escreva uma função que, dada uma tabela de inteiros e o seu número de elementos, coloca a 0 (zero) os três primeiros octetos (os mais significativos) dos inteiros nessa tabela. Comece por escrever um protótipo possível para a sua função e só depois passe à sua definição.

   (c) Escreva uma função `main` que mostre os valores na tabela antes e depois de modificada pela função implementada.

TPC Defina a seguinte estrutura:

```
struct carro {
   string modelo;     // modelo do carro  (mais do que uma palavra)
   string fabricante; // nome do fabricante (mais do que uma palavra)
   unsigned int ano;  // ano do carro
   float custo;       // custo do carro sem IVA
};
```

que pode armazenar o modelo, fabricante, ano e custo de um carro.

Considere que quer armazenar no máximo 300 carros.

   (a) Implemente uma função (le_carro) que pede ao utilizador que introduza o modelo, fabricante, ano e custo de um carro. A função recebe uma referência para um `struct carro` onde coloca (nos membros adequados) os valores dados pelo utilizador; considere o seguinte protótipo:

   ```
   void le_carro(carro & um);
   ```

   Questão: O protótipo `void le_carro(carro um);` seria uma alternativa viável ao anterior?

   Nota: Use uma chamada do tipo "`getline(cin, modelo_carro);`" para ler o modelo e fabricante de cada carro.

   (b) Implemente uma função, que utilizando a função `le_carro` implementada na alínea anterior, preencha uma tabela de `carro`, de capacidade máxima, `MAX_DIM` igual a 300. A função devolve o número de elementos efectivamente carregados.

   (c) Escreva uma função (`mostra`) que dada a tabela de `struct carro`, o número de elementos dessa tabela com informação introduzida, apresenta essa informação no ecrã ao seu gosto, mas utilizando uma linha para cada carro.

   (d) Escreva uma função que calcula o valor total (sem IVA) dos carros armazenados.

   (e) Escreva um programa principal que lhe permita testar estas três últimas funções.

Referências

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