DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Programação de Computadores (2013-2014)

Décima Primeira folha de Problemas

Matéria abordada: acesso a ficheiros, funções (estruturas; passagem de parâmetros por valor e por referência; argumentos na linha de comando; ponteiros.)
Bibliografia: Capítulos 12 e 16 de [Oua03].
Nota: Da folha 6 em diante é proibido o uso de variáveis globais!

1. Considere o problema 2 da folha 9 (se não concluiu a sua resolução na aula anterior, faça-o agora). Copie o ficheiro com resolução desse problema e designe-o por `f11e1.cpp`.

   Crie um programa que peça ao utilizador o nome de pessoas, o respectivo número de bilhete de identidade – BI (unsigned int) – e ainda a sua altura em metros. Utilize para esse efeito a função `carrega` implementadas na aula anterior.

   Seguidamente, e uma vez recolhida toda a informação, salve-a num ficheiro (cujo nome solicitará ao utilizador), colocando em cada linha a informação relativa a cada pessoa, de acordo com o formato que seguidamente se descreve. O ficheiro criado deverá possuir uma linha de título e a informação deve estar em colunas (alinhadas), sendo a altura especificada com duas casas decimais:

   ```
   Nome;                                BI        ;  Altura
   Manuel Carlos Amaral Monteiro Gomes; 123456789;    1.75
   Joaquim Xavier Domingos Martins;     98765432 ;    1.97
   ```

   Em que "BI" deverá estar na coluna 45, com os números dos BIs alinhados à esquerda nessa coluna; "Altura" deve estar à esquerda da coluna 75 (inclusive), e a altura deve estar alinhada à direita na coluna 75 em notação decimal com ponto fixo.

   Considere que o número máximo de valores que será possível introduzir está definido numa constante `MAXDIM` à qual atribuirá um valor (maior do que 0) à sua escolha. Considere que cada nome poderá ter no máximo 41 caracteres úteis – defina a constante `MAXLETRAS` e atribua-lhe o valor 41.

   Nesse programa utilizará (além da função `le()` implementada na aula anterior) as funções `le_nomeficheiro()`, `guarda_dados()` e `mostra()`, que seguidamente se descrevem:

   - A função `le_nomeficheiro()` deve solicitar ao utilizador um nome para o ficheiro, e não deverá aceitar nomes de comprimento nulo nem de comprimento superior a oito caracteres. Além disso só deve aceitar nomes formados apenas por letras (a-z, A-Z), e garantir que a primeira letra do nome é uma maiúscula, enquanto as restantes são minúsculas. Seguidamente acrescentará o sufixo ".txt" ao nome dado pelo utilizador. O nome do ficheiro será o único parâmetro dessa função.
   - A função `guarda_dados()` que, dada uma tabela de `struct ID` com os dados de um conjunto de cidadãos, o número de elementos nessa tabela com informação carregada, e o nome de um ficheiro, escreve toda a informação nesse ficheiro (se não existir cria um novo ficheiro, se existir apaga a informação nele contida e salva a informação dada).

     A função `guarda_dados()` deve ainda retornar *true* se a informação foi salva no ficheiro com sucesso e *false* caso contrário.

     Recorde que:

```
struct ID {
    string nome;            // nome da pessoa
    unsigned int BI;        // numero de BI da pessoa
    float altura;           // altura da pessoa em metros
};
```

Pode verificar a criação do ficheiro abrindo-o com o `kate` ou o `geany`, ou fazendo `cat` *dados.txt* na linha de comandos de um terminal.

- Escreva uma função, `mostra()`, que dado o nome de um ficheiro, ecoa para o ecrã o seu conteúdo (linha a linha).

  Sugestão: Leia cada linha para uma `string` que depois envia para `cout`. Use a função global `getline(ifstream &, string &);`

Sugestão: Escreva um programa principal que lhe permita testar a função `guarda_dados()`: utilize a função `mostra()` escrita na aula anterior para ecoar para o ecrã os valores armazenados na tabela e a nova função `mostra()` para mostrar o conteúdo do ficheiro, verificando se têm a mesma informação.

2. Copie a sua resolução do problema anterior para f11e2.cpp.

   Escreva um nova versão de `le()`, que dado o nome de um ficheiro – que deverá conter informação tal como está descrita no problema anterior – coloca toda a informação (com excepção da linha de título) numa tabela `struct ID`. O nome do ficheiro, a tabela e a dimensão máxima deverão ser os três parâmetros dessa função.

   A função deverá devolver -1 se houve algum erro no acesso ao ficheiro, caso contrário deverá devolver o número de elementos carregados na tabela.

   Escreva um programa principal que lhe permita testar a sua função – utilize a função `mostra()` escrita na aula anterior para ecoar para o ecrã os valores armazenados na tabela.

3. Crie um ficheiro `f11e3.h` onde coloque as definições das constantes, da `struct ID` e os protótipos da funções construídas - não de esqueça de incluir as directivas que impedem a múltipla inclusão deste ficheiro.

   Crie um ficheiro `bib10e3.cpp` onde coloque as definições das funções.

   Crie um ficheiro `f11e3.cpp` com o programa que escreveu no problema anterior.

   Compile separadamente `bib10e3.cpp` e `f11e3.cpp`. Crie o executável e teste-o.

4. Escreva um programa que replica o comportamento do comando "cat" no sistema linux ou "type". Designe-o por meucat.

5. As variáveis de tipo ponteiro terão particular importância na disciplina Estruturas de Dados e Algoritmos, pelo que é importante a sua compreensão.

   Diga quais as saídas resultantes dos seguintes pedaços de programa, sem implementar o respectivo código:

   (a)
   ```
   int i = 7;
   int * i_ptr = &i;
   *i_ptr = 5;
   std::cout << "\ni = " << i << "   *i_ptr = " << * i_ptr;
   ```

(b)
```
float x = 7.5;
float * x_ptr = &x;
x += 1;
std::cout << "\nx = " << x << "   *x_ptr = " << * x_ptr;
```

(c)
```
float t[3] = {4, 1.2, 3.4};
float * t_ptr = t;
for (int i=0; i < 3; i++, ++t_ptr)
   std::cout << "\n*t_ptr = " <<  * t_ptr;
```

(d)
```
char frase[] = "Made in EE.";
char *frase_ptr = frase;
std::cout << "\n    frase = " << frase;
std::cout << "\nfrase_ptr = " << frase_ptr;
```

(e)
```
char frase[] = "Made in EE.";
char *frase_ptr = frase;
for (int i = 0; frase[i] != '\0'; i++)
   std::cout << "\nfrase_ptr[" << i << "] = " << frase_ptr[i];
```

(f)
```
char texto[] = "Made in EE.";
char *texto_ptr = texto;
for (int i = 0; texto[i] != '\0' && i < 4; i++, texto_ptr++)
   std::cout << "\n*texto_ptr = " << *texto_ptr;
std::cout << std::endl;
```

6. Sem implementar o respectivo código, responda às seguintes questões.

   (a) Seguem-se dois pedaços de programa do mesmo tipo, um sobre tabela de inteiros, o outro sobre um tabela de caracteres. Diga quais as saídas resultantes.

   i.
```
int t[] = {4, 1, 5, 6};
int * t_ptr = t;
int dim = sizeof(t)/sizeof(int);
for (int i = 0; i < dim; ++i) {
   std::cout << "\n t[" << i << "] = " << t[i];
   std::cout << "\nt_ptr[" << i << "] = " << t_ptr[i];
}
```

   ii.
```
char t[] = "Ola.";
char * t_ptr = t;
int c = strlen(t);
for (int i = 0; i < c; ++i) {
   std::cout << "\n t[" << i << "] = " << t[i];
   std::cout << "\nt_ptr[" << i << "] = " << t_ptr[i];
}
```

(b) Seguem-se dois pedaços de programa do mesmo tipo, um sobre tabela de inteiros, o outro sobre um tabela de caracteres. Diga quais as saídas resultantes.

i.
```
int t[] = {4, 1, 5, 6};
int * t_ptr = t;
int dim = sizeof(t) / sizeof(int);
for (int i=0; i < dim; ++i, ++t_ptr)
   std::cout << "\n*t_ptr = " << *t_ptr;
```

ii.
```
char t[] =  "Ola.";
char * t_ptr = t;
int c = strlen(t);
for (int i = 0; i < c; ++i, t_ptr++)
   std::cout << "\n*t_ptr = " << *t_ptr;
```

(c) Diga quais as saídas resultantes.

```
char t[] =  "Ola.";
char * t_ptr = t;
int c = strlen(t);
for (int i = 0; i < c; ++i, ++t_ptr)
   std::cout << "\nt_ptr = " << t_ptr;
```

(d) Diga quais as saídas resultantes.

```
char t[] =  "Ola.";
char * t_ptr = &t[1];
std::cout << "\nt_ptr = " << t[1];
std::cout << "\nt_ptr = " << &t[1];
std::cout << "\nt_ptr = " << t_ptr;
std::cout << "\nt_ptr = " << t_ptr[0];
std::cout << "\nt_ptr = " << *t_ptr;
t_ptr++;
std::cout << "\nt_ptr = " << *t_ptr;
```

Ajuda para os problemas 1 e 2:

- Para criar um ficheiro de saída pode fazer:

```
std::ofstream escreve1(nomefich);
```

em que `nomefich` deve ser do tipo `char *` e apontar para um nome de ficheiro válido. Por omissão, ou seja sem activar a bandeira `std::ios::binary`, o ficheiro criado será ASCII.

- Para abrir um ficheiro de entrada pode fazer:

```
std::ifstream leitura(nomefich);
```

em que `nomefich` deve ser do tipo `char *` e apontar para um nome de ficheiro válido.

- O manipulador `setprecision` define o número de casas decimais à direita da vírgula. Para que o formato (número de casas decimais) seja respeitado mesmo quando o valor a mostrar

é "2", terá de activar a bandeira `ios::fixed` Para activar a bandeira `ios::fixed` use a função `setf`, função membro da classe `ofstream`.

Para as restantes opções, consulte o formulário!

Referências

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