Ficha Prática nº4

Estruturas, Apontadores e Ficheiros

Programação III

Paul Crocker e Simão Melo de Sousa
Departamento de Matemática/Informática da UBI

23 de Março de 2001

1 Registos

Exercício 1:
Escreva uma função que aceite um estrutura do tipo `LINE` e devolva um inteiro (ou 1, ou 2, ou 3), onde 1 indica uma linha vertical, 2 indica uma linha horizontal e 3 uma linha oblíqua. Uma linha vertical é uma linha que tem as coordenadas x iguais, uma horizontal tem as coordenadas y iguais e uma linha oblíqua é a que não é vertical nem horizontal.

```
typedef struct { int x; int y; } POINT;
typedef struct { POINT beg;  POINT end;  } LINE;
```

---

2 Apontadores

Exercício 2:
Escreva a função seguinte que recebe um float e devolve a parte inteira e a parte fraccional.
`void intFraction (float num, int *integer, float *fraction)`

Exercício 3:
Dado dois arrays de `N` inteiros, `A` e `B`. Escreva uma função que verifique se os elementos do `A` são iguais aos elementos do array `B`. Ou seja para todo o $i \in \{0 \cdots N-1\}$ $A[i] = B[i]$. A função deverá devolver zero no caso de igualdade, um no caso contrário.
`int checkData (int *pAry1, int *pAry2, int tamanho)`

Exercício 4:
Escreva uma função que inverta a posição dos elementos dum vector (ou seja, o primeiro elemento torna-se o último etc...).
A função deverá aceitar um apontador para o vector e um inteiro para representar o tamanho do vector.

Exercício 5:
Escreva uma função que verifique um código do International Standard Book Number (ISBN) para ver se é valido ou não. O número ISBN é usado para definir unicamente um livro. É composto por 10 elementos que são dígitos, menos o último que também poderá ser um x (que representa o valor 10). Por exemplo 123456789x ou 1122334455. Um número é valido se a soma dos dígitos multiplicados pelo seu peso for divisível por 11.

Requisitos: isbn: vector de 10 inteiros
Para i **A Variar de** 0 **Até** 9 **Fazer**
    soma = soma + (valor(isbn[i]) * (i + 1))

A função deverá aceitar um apontador para uma string de caracteres e devolver um inteiro, 0 para não-valido e 1 para valido.

Exercício 6:
Dada a seguinte declaração e definição : `char a[6] = {'z', 'x', 'm', 's', 'e', 'h'} ;`
Utilizando exclusivamente a notação apontadores, escreva um ciclo para puxar todos os valores de uma posição para a direita. Deverá imprimir os caracteres antes e depois da mudança.

```
#include <stdio.h>
int main (void)
{
/* Local Declarations */
int ary[6] = {'z', 'x', 'm', 's', 'e', 'h'};
...
O seu código
```

Exercício 7: (*Função Multiuso*)
Escreva uma função que aceita um vector de inteiros e que devolva a média dos valors presentes, a soma dos valores, o maior valor e o menor valor e que retorna um valor de erro (por exemplo -1, 0 senão) se vector não estiver ordenado.

Exercício 8:

1. Escreva uma função para trocar dois floats com o seguinte protótipo

   ```
   void trocar(float *a, float *b);
   ```

   Escreva um programa (a função main) que teste a função e imprime os valores e endereços das variaveis envolvidas.

2. Escreva um programa que leia duas variaveis do tipo POINT e que troca os conteúdos. A função deverá utilizar as seguintes três funções.

   ```
   void trocar(POINT *x, POINT *y);
   void imprimir(POINT x);
   void iniciarponto(POINT *x);
   ```

Exercício 9: (*Strings como Apontadores*)
Reescreva as funções do exercício **10** da ficha anterior considerando uma string como um apontador para caracteres.

---

3 Argumentos do Main

Exercício 10: (*Programa Variável*)
Escreva um programa que:

- Se se chamar `ordem` devolve a maior palavra (usando a ordem alfabética) dos argumentos fornecidos. Tomar esta opção como sendo a opção por defeito;
- Se se chamar `conta` devolve o número de argumentos;
- Se se chamar `lista` manda para a saída standard os seu argumentos.

Exercício 11: (*Word Count*)
Escreva um programa C que simula o funcionamento do comando `wc` do sistema Unix com a restrição que a entrada de caracteres utilizada é a entrada standard e que as opções aceites são `-l -w -c` para respectivamente "só as linhas", "só as palavras"e "só os caracteres".

- Extende o seu programa para que possa aceitar argumentos do tipo ficheiro a semelhança do comando `wc` do unix.

---

4 Memória Dinamicamente Reservada: (malloc, calloc e free)

Exercício 12:
O objectivo desta exercicio é escrever uma função que simplesmente soma dois inteiros !

1. Implemente primeiros os dois casos simples:

   ```
   int soma1 ( int a, int b);
   int soma2 (int *a, int *b);
   ```

2. Escreva a seguinte função `soma3` (errada !), e explique o seu indesejado comportamento utilizando se necessário exemplos de execução.

   ```
   int * soma3 (int a, int b);
   {
         int  temp;
         temp = a+b;
         return &temp;
   }
   ```

3. Utilizando malloc, escreva

   ```
   int * soma4(int a, int b);
   int * soma5(int *a, int *b);
   ```

4. Utilizando uma variável static.

   ```
   int * soma6 ( int a, int b);
   int * soma7(int *a, int *b);
   ```

Exercício 13: (*Vectores de Strings*)

1. Defina o tipo dos vectores de N apontadores para strings (onde N é uma constante);
2. Defina a função que permita inicializar um tal vector;
3. Defina uma função que permita mostrar o i-ésimo elemento;
4. Defina uma função que permita ordenar por ordem lexicográfica (ou seja alfabética) as strings do vector.
5. Redefina as alíneas anteriores de tal forma que seja possível trabalhar com um vector de apontadores de tamanho qualquer.

---

5 Ficheiros

Exercício 14: (*Manipulação de Ficheiros*)

- Escreva:
  1. Uma função que conta os caracteres, palavras e linhas dum ficheiro;
  2. Uma função que permita copiar um ficheiro para outro;
  3. Uma função que concatena um ficheiro outro.

Exercício 15: (*Processamento de texto*)
Escreva um programa que permita construir uma lista de todas as palavras contidas num ficheiro, assim como a occorência de cada uma delas

---

Exercício 16:
Escreva um programa para criar um ficheiro binário de inteiros. O seu programa deverá pedir o nome (max 20 caracteres) do ficheiro para escrever.

Exercício 17:
Escreva uma função que copia o conteúdo dum ficheiro binário para um segundo ficheiro. A função deverá aceitar dois apontadores para ficheiros e devolver um inteiro (0 em caso de erro de processamento, 1 em caso de sucesso).

Exercício 18:
Escreva umas função que compara dois ficheiros e devolva 0 se são iguais, 1 no caso contrário. A função deverá receber apontadores para dois ficheiros previamente abertos e compará-los byte por byte.

Exercício 19:
Escreva uma função que imprime o último inteiro dum ficheiro binário de inteiro.

Exercício 20:
Escreva um programa para criar um ficheiro binário cujo dados são do tipo


```
typedef struct  {   int  data;   char c; } STR;
```

Exercício 21:
Escreva uma função que concatena no fim de um ficheiro binário outro ficheiro binário.

Exercício 22:
Escreva uma função que lê itens de um ficheiro binário e que os copia para um vector "dinámico"(cujo tamanho foi estabelecido após o conhecimento do tamanho do ficheiro binário, logo utiliza apontadores e alocação de memória).

---

6 Projectos

Exercício 23: (*Listas Ligadas de Inteiros*)
Colocamo-nos neste exercício no caso das listas ligadas de inteiros. uma lista ligada é de forma geral definida por dois tipos: o tipo `info` que representa a informação arquivada na lista, ou seja, no nosso caso `info` é sinónimo de `int`, e o tipo `nodo` que representa a forma como uma informação vai estar ligada às outras. `nodo` representa assim a coluna vertebral da lista ligada porque contem um campo com uma informação e um apontador para o nodo seguinte.

1. Defina os tipos necessários que definam as listas ligadas de inteiros;
2. Defina as seguintes funções de inserção de informação numa lista ligada de inteiros:
   (a) função de inserção ao início da lista;
   (b) função de inserção ao fim da lista;
   (c) função de inserção ordenada na lista;
3. Defina uma função de remoção de informação;
4. Defina uma função de listagem de informação;
5. Defina uma função que permita a procura um elemento na lista;
6. Defina uma função que permita calcular a média dos elementos da lista;
7. Defina uma função que permita a aplicação de uma função aos elementos da lista.

Exercício 24: (*Um Programa para a Boa Vida*)
Pretendemos, neste exercício, manter uma base de dados sobre os lugares de perdição da zona da Beira Interior. Pretendemos guardar para esse efeito as informações sobre os bares numa lista ligada. Para cada bar guardamos o seu nome, a cidade onde se encontra, a hora de fecho, a categoria do bar (café, pub, discoteca ...), o estilo de música (se houver), o preço da cerveja.

1. Defina os tipos adequados para que essa informação possa ser processada por uma lista ligada;
2. Defina uma função de inserção;
3. Defina funções de listagem: listagem geral e sob critérios (por exemplo uma função que mostra todos os bares que tem a cerveja mais barata que um preço introduzido pelo utilizador do programa, ou os bares que ainda estão abertos depois de uma hora definida mais uma vez pelo utilizador);
4. Defina uma função de procura por nome e outra por estilo musical;
5. Defina uma função de remoção por nome;
6. Defina as funções que achar úteis.

Exercício 25: (*Árvores Binárias de inteiros*)
Uma árvore binária é, como a lista ligada, definida a custa dos dois tipos: o tipo `info` e o tipo `nodo`. A diferência essencial é que um nodo não tem *um* sucessor, mas *dois*. Costuma-se chamar **filhos** aos sucessores, e **pai** ao predecessor de um nodo. O nodo que está ao topo da hierarquia não tem pai e é chamado **raíz**. Para um dado nodo, os dois filhos são classificados e chamados respectivamente **filho esquerdo** e **filho direito**. Um nódo que não tem filhos é chamado **folha**. A inserção de uma informação inf é geralmente descrita, de forma simples, recursivamente [1] nos seguintes termos: para uma dada posição p na árvore,

- se $p.info > inf^2$ então deslocamos o problema de inserção de p para o filho esquerdo de p;
- se $p.info < inf$ desloca-se então o problema de inserção para o filho direito.
- Se por acaso o filho que vai receber a tarefa de inserção não existir, deverá criar-se o dito filho com o conteúdo inf.

Tem-se assim uma estructura que arquiva as informações a medida que essas chegam, mas de forma ordenada sem passar por uma ordenação constante e cara em recursos.
A remoção utiliza as propriedades que garante uma tal inserção. Para qualquer nodo p, é garantido que não existe na sua sub-árvore esquerda um nódo com uma informação maior[3], assim como não existe na sub-árvore direita uma informação menor do que a informação que p arquiva.

[1] Apesar dos problemas de eficiência que uma tal implementação traz.
[2] $>$ é aqui utilizada para representar uma ordem estricta, que é representada por "maior"nos inteiros.
[3] Relembramos que a ordem utilizada não é necessariamente "maior", pode ser, no caso das strings, a ordem alfabética.

Colocamo-nos agora no caso geral da remoção. Imaginamos que pretendemos remover p. Seja q o pai de p. Fazemos agora a suposição de que p é o filho direito de q [4] [5]. Remover p consiste então em tirar p da árvore e de colar a sub-árvore esquerda de p na primeira posição livre a esquerda da sub-árvore direita de p (percorre-se a sub-árvore direita de p pela esquerda, o primeiro nodo que não tem filho esquerdo terá como sub-árvore esquerda a sub-árvore esquerda de p). O processo de remoção é finalmente concluído tornando a raíz da sub-árvore direita de p (já com a primeira colagem efectuada) filho direito de q.

1. Defina os tipos adequados que modelisam uma árvore binária de inteiros;
2. Defina uma função de inserção;
3. Defina um método de remoção para todos os casos e implemente-a através uma função C;
4. Defina uma função de listagem;
5. Defina uma função de procura.

Exercício 26: (*A Boa Vida em Árvore*)
Reescreva o programa de gestão da base de dados dos Bares usando as árvores binárias.

Exercício 27: (*Salvar os Bares da Covilhã*)
Modifica o programa "Bares"de tal forma que:

1. seja possível gravar a informação para o disco de tal forma que a informação de um bar seja gravada numa linha e que cada campo seja separado dos outro por uma tabulação.
2. seja possível ler a informação a partir do disco.

---

[4] O caso contrário é completamente simétrico, por isso só consideramos este caso.
[5] Os casos particulares são então o caso simétrico, o caso de p não ter filhos, ou só ter um dos dois filhos, o caso de p ser raíz.