Programação 2024/25

LEI, LEI-PL, LEI-CE

Aula Laboratorial 7

Bibliografia:

K. N. King. *C programming: A Modern Approach* (2nd Edition). W. W. Norton: capítulo 17.

Código de apoio para a aula: https://gist.github.com/FranciscoBPereira

Arrays Dinâmicos

Exercícios Obrigatórios

1. Considere o seguinte programa:

```
#include <stdio.h>
#include <stdlib.h>

void preenche(int *a, int tam){
    for(int i=0; i<tam; i++)
        a[i] = 5*i;
}

int* f1(int *a, int *tam){
    int *p, i;

    p = realloc(a, sizeof(int)*(*tam + a[1]));
    if(p == NULL)
       return a;
    a = p;
    preenche(a + *tam, a[1]);
    *tam += a[1];
    return a;
}

int main(){
    int *v = NULL, total = 0;

    v = malloc(sizeof(int) * 3);
    if(v == NULL)
        return 0;
    total = 3;
    preenche(v, total);
    v = f1(v, &total);
    free(v);
    return 0;
}
```

a) Identifique os locais exatos onde é efetuada gestão dinâmica de memória. O que acontece em cada um desses locais?

b) Em quais dos locais identificados é que podem existir erros de gestão de memória? O que faz o programa, caso esses erros aconteçam?

c) Qual a dimensão e conteúdo do vetor de inteiros imediatamente antes da chamada da função *free(),* assumindo que até esse momento não existiu nenhum erro de gestão de memória?

d) Qual a dimensão e conteúdo do vetor de inteiros imediatamente antes da chamada da função *free()*, assumindo que existiu um erro de gestão de memória na função *f1()*?

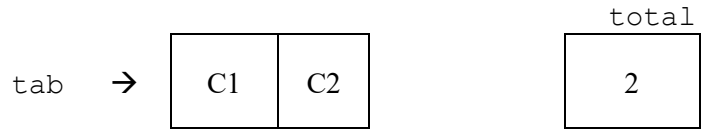
2. Pretende-se criar uma agenda para armazenar contactos de telemóvel (nome e número). A agenda não deve ter um tamanho máximo pré-definido, ou seja, terá a capacidade necessária e suficiente para armazenar os contactos existentes em cada momento.

a) Crie um tipo estruturado que lhe permita armazenar a informação de um contacto.

b) Declare as variáveis na função *main()* que lhe irão permitir gerir um array dinâmico de contactos telefónicos. Vai necessitar de um ponteiro para referenciar o array e de uma variável inteira para armazenar a dimensão da tabela.

c) Escreva uma função que adicione um novo contacto à agenda. Toda a informação necessária é indicada pelo utilizador. Deve garantir que os nomes são únicos, ou seja, não podem existir contactos com o mesmo nome na agenda. A função recebe o endereço do array dinâmico e o endereço da variável inteira contendo a dimensão do vetor. Devolve o endereço do vetor depois da atualização.

Exemplo: Após a adição de 2 clientes, a estrutura de dados terá a informação ilustrada na figura seguinte.



d) Escreva uma função que liste todos os contactos existentes na agenda. A função recebe o endereço e a dimensão do array dinâmico.

e) Escreva uma função que procure o número de telemóvel de um determinado contacto. A função recebe o endereço e a dimensão do array dinâmico e o nome do contacto a pesquisar. Devolve o número de telemóvel do contacto indicado, ou -1, no caso do contacto não existir.

f) Escreva uma função que atualize o número de telemóvel de um determinado contacto armazenado na agenda. A função recebe o endereço e a dimensão do array dinâmico, o nome do contacto a atualizar e o novo número de telemóvel. Devolve 1 se a atualização for efetuada com sucesso, ou 0, caso contrário.

g) Assuma que os 2 primeiros dígitos do número de telemóvel identificam o operador móvel a que pertence. Escreva uma função que descubra quantos operadores móveis diferentes existem nos contactos armazenados no array. A função recebe o endereço e a dimensão do array dinâmico. Devolve o valor contabilizado.

h) Escreva uma pessoa que elimine da agenda o contacto de uma pessoa. A função recebe o endereço do array dinâmico, o endereço da variável inteira contendo a dimensão do vetor e o nome da pessoa a eliminar. Devolve o endereço do vetor depois da atualização.

Sugestão: Escreva 2 versões desta função, uma utilizando o *realloc( )* e a outra sem recorrer a esta função.

i) Altere o que for necessário nas alíneas anteriores para garantir que os contactos na agenda se encontram ordenados pelo nome da pessoa.

Exercícios Complementares

3. Considere as seguintes estruturas:

```
typedef struct movimento mov;
typedef struct {int d, m, a;} data;
struct movimento{
      data dMov;
      char nconta[15];
      int val;
};
```

Programação 2024/25

LEI, LEI-PL, LEI-CE

Cada estrutura do tipo *mov* consegue armazenar um movimento realizado por um determinado cliente de um banco. O campo *dMov* indica a data em que foi realizada a operação, o campo *nconta* contém o identificador da conta que realizou o movimento e o campo *val* indica qual o valor movimentado (positivo para depósito ou negativo para levantamento).
Todos os movimentos realizados numa agência bancária ao longo de um período vão sendo armazenados num array dinâmico. De cada vez que um movimento é efetuado, uma nova estrutura é adicionada ao final do array.

a) Declare as variáveis na função *main()* que lhe irão permitir gerir um array dinâmico de movimentos.

b) Escreva uma função que adicione um novo movimento ao array. Toda a informação necessária é indicada pelo utilizador. O novo movimento deve ser colocado no final do array.

c) Escreva uma função que liste todos os movimentos existentes no array.

d) Escreva uma função que liste todos os movimentos do array que foram realizados num determinado período. As datas limite do período são dois dos parâmetros da função.

e) Escreva uma função que, dado um número de conta, devolva a seguinte informação: número total de movimentos realizados, número de levantamentos, número de depósitos, saldo acumulado dos movimentos realizados.

f) Escreva uma pessoa que elimine do array todos os movimentos efetuados numa determinada data. A data a considerar é um dos parâmetros da função.

Programação 2024/25

LEI, LEI-PL, LEI-CE

Trabalho Autónomo

4. Implemente um programa em C que efetue a gestão de um pequeno ecossistema consistindo num conjunto de peixes que habita alguns aquários. A definição deste problema é propositadamente vaga e deverá completá-la de uma forma razoável e que não entre em conflito com o que é explicitamente referido:
- O ecossistema contém vários aquários. Cada aquário é definido, no mínimo, através dos seguintes atributos: Nome, identificador único, capacidade máxima (número de peixes que pode conter) e capacidade atual (número de peixes que contém nesse momento).
- O ecossistema contém vários peixes. Cada peixe é definido, no mínimo, através dos seguintes atributos: Espécie, identificador único, peso e identificador do aquário onde se encontra nesse momento. Não podem existir peixes no ecossistema que não se encontrem num dos aquários.
O programa deve permitir efetuar as seguintes operações:

a) Adicionar um novo aquário. Quando é adicionado, o aquário fica sem peixes.

b) Eliminar um aquário. O aquário só poderá ser eliminado do ecossistema se não tiver peixes.

c) Listar todos os aquários.

d) Listar a identificação de todos os peixes que se encontrem num determinado aquário.

e) Adicionar um novo peixe ao ecossistema. O peixe deve ficar obrigatoriamente num dos aquários que já existem

f) Eliminar um peixe do ecossistema.

g) Listar todos os peixes que se encontram no ecossistema. Para cada peixe, esta listagem deve incluir a identificação do aquário em que se encontram.

h) Transferir um peixe de um aquário para outro.