Algoritmos e Estruturas de Dados I (DCC/003)

Aulas Práticas – Parte 2 do curso
Data de entrega: 06-07-2017

Orientações gerais:

- Os exercícios deverão ser preferencialmente feitos durante o tempo das aulas de laboratório;
- As dúvidas serão esclarecidas nas aulas;
- Os exercícios deverão ser entregues via Moodle;
- Os executáveis não devem ser enviados, apenas os arquivos de extensão ".c".

Ponteiros e Alocação Dinâmica

1) Faça uma função que retorne um array de N inteiros alocados dinamicamente. Esse array deve ser preenchido com valores de 0 a N-1.

[salve o seu código com o nome: **ap01ex1.c**]

2) Escreva um programa que leia dois números inteiros, m e n, e duas matrizes A e B, de dimensões m por n, e imprima a soma. Você não pode utilizar vetores ou matrizes estáticos, ou seja, deve usar **alocação dinâmica**, da forma que melhor desejar. Após a impressão do resultado você deve **desalocar** todo o espaço utilizado.

A entrada será dada por uma linha, contendo dois inteiros, m e n. Em seguida, m linhas conterão, cada uma, n números inteiros, referente ao conteúdo da matriz A. Finalmente, as m linhas seguintes conterão n inteiros cada, correspondendo ao conteúdo de B.

Exemplo de Entrada 1:

```
1 1
1
2
```

Exemplo de Saída correspondente à Entrada 1:

```
3
```

Exemplo de Entrada 2:

```
2 5
1 2 3 4 5
0 0 0 0 0
2 3 4 5 6
1 0 1 0 1
```

Exemplo de Saída correspondente à Entrada 2:

```
3 5 7 9 11
1 0 1 0 1
```

[salve o seu código com o nome: **ap01ex2.c**]

3) Dado um array com os valores 25, 4 ,10, 30, 20, 1, 3, 5, 6. Implemente uma função que modifique esse array para que a saída seja um vetor onde a posição i contenha a soma de todos os valores de 0 a i do vetor de entrada. Imprima o vetor resultante na

função principal.

[salve o seu código com o nome: **ap01ex3.c**]

4) Escreva um procedimento de nome aumentaOsIguais que recebe como parâmetro dois endereços de memória de variáveis inteiras end_var1 e end_var2. A função deve verificar se esses endereços de memória têm o mesmo valor inteiro armazenado neles. Caso negativo, a função deve subtrair 1 de ambos conteúdos dos endereços de memória. Caso positivo, a função deve fazer a soma dos dois valores e armazenar essa soma em ambos endereços de memória.

[salve o seu código com o nome: **ap01ex4.c**]

Registro - Struct

1) Vamos elaborar um programa em C para praticar os conceitos de estrutura ou registro – *struct.*

O que deverá ser feito é descrito a seguir:

Um hospital da rede pública de saúde deseja fazer o cadastro dos seguintes dados sobre os pacientes que atende: **nome**, **idade**, **peso** e **altura**. Defina uma estrutura de dados conveniente para armazenar estes dados (*struct*).

```
typedef struct pessoa
{
        ...
        ...
} individuo;
```

Considere que o cadastro deverá ser armazenado em um **vetor** (denominado **vetorCadastro**), sendo que o vetor deverá ter um tamanho máximo permitido (#define MAX 10).

No início do código principal, pergunte ao usuário quantas pessoas serão cadastradas (n). Para cada pessoa a ser cadastrada, você receberá os dados e colocará no vetor do tipo da estrutura que você criou (individuo).

Em seu código **main** você deverá ter:

```
int main (ints argc, char *argv[])
{
        individuo vetorCadastro[MAX];

        faça um estrutura de repetição para receber os cadastros das pessoas.

        faça um FOR (para cada pessoa), imprimindo seus campos:
                - vetorCadastro[i].nome;
                - vetorCadastro[i].altura;
                - etc.

}
```

[Salve o código como **ap02-ex01.c**.]

2) Faça uma **função** chamada **nova_pessoa()** para receber o cadastro de cada pessoa e retornar um tipo **individuo** (do *struct* que você criou).

```
individuo nova_pessoa()
```

```
{
        individuo p;
        ...
        Leia os dados
        ...
        Retorne o individuo
}
```

E altere seu código **main** implementado anteriormente para que faça a chamada à função que vai gerando o registro de cada novo indivíduo.

[Salve o código como **ap02-ex02.c**.]

3) Escreva um programa que crie um tipo de dado de enumeração para os dias da semana. Em seguida, solicite ao usuário para digitar um número equivalente a um determinado dia da semana, compare com o número no tipo de dado de enumeração e exiba o nome do dia da semana na tela.

[Salve o código como **ap02-ex03.c**.]

Arquivos

1) Vamos trabalhar agora com um novo conceito, o de arquivos em C, usando novos comandos, que serão importantes em outros exercícios futuros.

Elabore um programa que realiza a leitura de um arquivo texto e trabalhe com os dados do arquivo para imprimir informações e gerar média dos dados informados.

Salve o código como **ap03-ex1.c**.

Para ajudar a elaborar o programa, siga os seguintes passos:

Inclua as bibliotecas necessárias e defina uma constante MAX:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>   // necessário para algumas operações com arquivos
#include <string.h>  // necessário para strtok()
#include <ctype.h>

#define MAX 80       // maximo de caracteres a serem lidos
```

Agora crie uma função que represente o MENU de opções de seu programa:

```
char opcao_menu()
{
  system("cls");
  printf(" (L)istar notas\n");
  printf(" (F)im\n");
  printf("> ");
  return (toupper(getche()));
}
```

Agora vamos criar outra função que faça listagem das notas que serão lidas do arquivo de entrada (que chamaremos de dados.txt):

```
void listar_notas()
{
  int num,notas;
  float n1,n2,media;
  char *nome;
  char buf[MAX];
```

```
  FILE *arq;

  arq = fopen("dados.txt","r");
  if (arq == NULL)
  {
    printf("Erro ao abrir arquivo\n");
    return;
  }
  printf("\n");
  printf("NUM |         NOME          |  N1  |  N2  \n");
  printf("----+-----------------------+------+------\n");

  notas = 0;
  media = 0;

  fgets(buf,MAX,arq);
  while (!feof(arq))
  {
    num = atoi(strtok(buf,","));
    nome = strtok(NULL,",");
    n1 = atof(strtok(NULL,","));
    n2 = atof(strtok(NULL,","));
  printf("%03d | %-20s | %4.1f | %4.1f\n",num,nome,n1,n2);
    notas = notas + 2;
    media = media + n1 + n2;
    fgets(buf,MAX,arq);
  }
  printf("----+-----------------------+------+------\n");
  media = media/notas;
  printf("Media das notas = %4.1f\n",media);
  fclose(arq);
}
```

Atenção aos seguintes novos conceitos:

*arq → é um ponteiro para o arquivo de onde os dados serão lidos;
arq = fopen("dados.txt","r") → forma de ler dados do arquivo, retornando na variável com ponteiro para o arquivo;
fgets(buf,MAX,arq)→ commando para ir lendo do arquivo (cada linha);
atoi → converte char para inteiro;
atof → converte char para float (real);
strtok → vai quebrando a linha lida, com separador informado (aqui estamos usando vírgula como separador, que será colocado no arquivo de dados);
fclose → comando para fechar o arquivo.

Agora vamos criar o código principal deste exercício:

```
int main(int args, char * arg[])
{
  char op;

  do
  {
    op = opcao_menu();
    if (op == 'L')
      listar_notas();
    printf("\n");
    system("pause");
  }
  while (op != 'F');
  return 0;
```

```
}
```

Para testar, crie um arquivo chamado “dados.txt” com o seguinte conteúdo:

```
2,Maria da Silva,8.5,4.8
13,Joao de Almeida,7.5,6.1
15,Jose de Almeida,8.8,9.3
21,Maria Joao,7.3,7.3
```

2) Faça um programa que multiple 2 matrizes **n** x **m** e **m** x **p**. A matriz resultante deve ser impressa na tela.

Teste para o seguinte arquivo de entrada (chame-o de matrizes.txt):

```
3 4
1 2 3 4
4 3 2 1
2 3 4 1

4 2
2 3
5 2
3 1
3 3
```

O arquivo acima contem as dimensões da primeira matriz (**n** x **m**), os valores da primeira matriz, as dimensões da segunda matrix (**m** x **p**) e os valores da segunda matriz. Para o exemplo acima, o resultado da multiplicação da matrix acima será:

```
33 22
32 23
34 19
```

[Salve o código como **ap03-ex2.c**.]

3) Crie uma estrutura que represente um aluno(nome, idade e matricula). Utilizando essa estrutura, escreva um programa que leia os dados de 5 alunos e os armazene em um arquivo binário.

[Salve o código como **ap03-ex3.c**.]