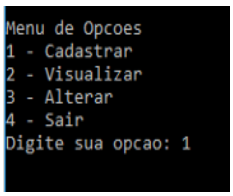
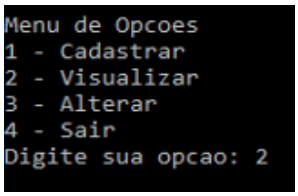
Correção dos exercícios da aula anterior

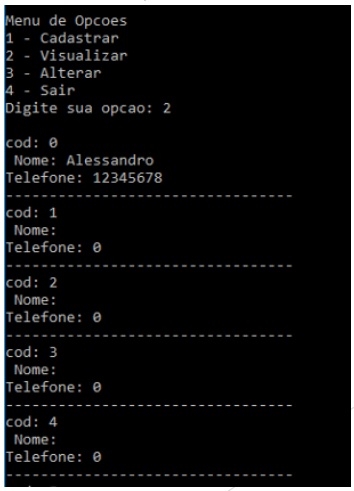
1- Crie uma agenda telefônica com 15 registros. Utilize as imagens a seguir para referência do que deve ser programado.

Cadastrar:

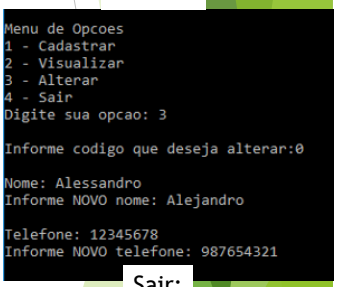


Visualizar: OBS: Mostra todos os espaços da agenda.

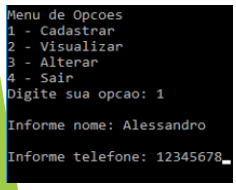


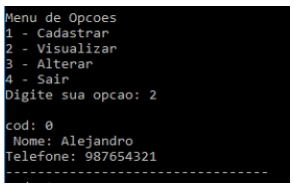


Alterar:

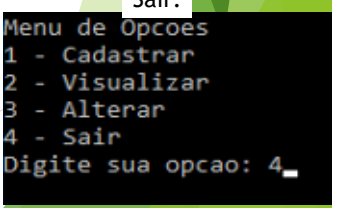


Visualizar: o que foi alterado





Sair:



1-(1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
//Declaracao do registro
struct {
char nome[35];
int telefone;
}  agenda[15];

//Funcao Principal
main()
{
struct agenda;
    int op, i=0, alt;
    do
    {
    system("cls");//limpar tela
    printf("\nMenu de Opcoes");
    printf("\n1 - Cadastrar ");
    printf("\n2 - Visualizar");
    printf("\n3 - Alterar");
    printf("\n4 - Sair");
    printf("\nDigite sua opcao: ");
    scanf("%d",&op);
        switch(op)
        {
        case 1:
            printf("\nInforme nome: ");
            scanf("%s",agenda[i].nome);
            printf("\nInforme telefone: ");
            scanf("%d",&agenda[i].telefone);
            i++;
        break;
```


1-(2/2)

```c
        case 2:
        for(i=0;i<15;i++)
        {
            printf("\ncod: %d \n Nome: %s",i,agenda[i].nome);
            printf("\nTelefone: %d",agenda[i].telefone);
            printf("\n---------------------------------");
        }
        getch();
        break;
        case 3:
            printf("\nInforme codigo que deseja alterar:");
            scanf("%d",&alt);
            printf("\nNome: %s",agenda[alt].nome);
            printf("\nInforme N OVO nome: ");
            scanf("%s",agenda[alt].nome);
            printf("\nTelefone: %d",agenda[alt].telefone);
            printf("\nInforme NOVO telefone: ");
            scanf("%d",&agenda[alt].telefone);
        break;
        case 4:
        break;
        default:
            printf("\nOpcao invalida");
        break;
        }
    }while(op!=4);
    getch();
}
```

Exercício Complementar

2- Desenvolva um programa na linguagem C que cadastre o nome, a altura, o peso, o cpf e o sexo (M para masculino e F para feminino) de uma pessoa.

Antes do cadastro, pergunte ao usuário quantos registros ele deseja realizar.

Com os dados cadastrados, em seguida localize uma das pessoas por meio do seu CPF e imprima o seu IMC em tela.

OBS: Calculo do IMC = peso / (altura*altura)

Exercício Complementar(1/2)

2-

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

struct Pessoa{
        char  nome[100];
        char  sexo; // 'M': masculino, 'F': feminino
        float peso;
        float altura;
        long cpf;
};
main()
{
        int i=0, j=0, temp=0, n_cadastrado=0;
        long cpf_localizador;
        float imc=0;
        struct Pessoa pessoas[15];

        printf("Informe o numero de cadastrado que deseja realizar ");
        scanf("%d", &n_cadastrado);

        for(i=0; i<n_cadastrado; i++)
        {
                printf("\nInforme o seu nome (%i): ",i+1);
                scanf("%s",pessoas[i].nome);
                fflush(stdin);
                printf("\nInforme a sua altura (%i): ",i+1);
                scanf("%f",&pessoas[i].altura);
                fflush(stdin);
                printf("\nInforme o seu peso (%i): ",i+1);
                scanf("%f",&pessoas[i].peso);
                fflush(stdin);
                printf("\nInforme o seu CPF (%i): ",i+1);
                scanf("%lu",&pessoas[i].cpf);
                fflush(stdin);
```

Exercício Complementar(2/2)

2-

```
36              printf("\nInforme o seu sexo, M para Masculino, e F para Feminino (%i): ",i+1);
37              scanf("%c",&pessoas[i].sexo);
38              fflush(stdin);
39          }
40          system("cls");
41          printf("\nInforme o CPF da pessoa a ser localizado: ");
42          scanf("%lu",&cpf_localizador);
43          fflush(stdin);
44
45          for(j=0; j < n_cadastrado; j++)
46          {
47              if (cpf_localizador==pessoas[j].cpf)
48                  {
49                      printf("\nSexo\tNome\tIMC");
50                      imc = pessoas[j].peso/(pessoas[j].altura*pessoas[j].altura);
51                      printf("\n%c\t%s\t%1.2f\n",pessoas[j].sexo, pessoas[j].nome, imc);
52                      break;
53                  }
54          }
55          getch();
56  }
```


Algoritmos e Práticas de Programação

Prof. Alessandro Santos

Ponteiros

Ponteiros

- Toda informação que é manipulada dentro de um programa (esteja ela guardada em uma variável, *array,* estrutura, entre outros) obrigatoriamente está armazenada na memória do computador.
- Quando é criada uma variável, o computador reserva um espaço de memória para guardar o valor associado a essa variável.
- O nome que é dado a essa variável o computador associa o endereço do espaço que ele reservou na memória para guardar essa variável.
- De modo geral, interessa ao programador saber o nome das variáveis. Já o computador precisa saber onde elas estão na memória, ou seja, precisa dos endereços das variáveis.

O que são Ponteiros?

- Ponteiros são um tipo especial de variáveis que permitem armazenar endereços de memória ao invés de dados numéricos (como os tipos **int**, **float** e **double**) ou caracteres (como o tipo **char**).

Qual é a diferença entre uma Variável e um Ponteiro?

- • Variável: é um espaço reservado de memória usado para guardar um valor que pode ser modificado pelo programa;
- • Ponteiro: é um espaço reservado de memória usado para guardar um endereço de memória.

Declaração

- Em linguagem C, a declaração de um ponteiro segue a seguinte sintaxe:
- tipo do ponteiro *nome do ponteiro;
- Exemplo:
- int *p;
- float *x;
- char *y;

Os Operadores "*" e "&"

- Ao se trabalhar com ponteiros, duas tarefas básicas serão sempre executadas:

- Acessar o endereço de memória de uma variável;
- Acessar o conteúdo de um endereço de memória;

- Para realizar essas tarefas, é necessário sempre utilizar apenas dois operadores: o operador "*" e o operador "&".

Os Operadores "*" e "&"

Operador "*" versus operador "&"	
"*"	Declara um ponteiro: int *x; Conteúdo para onde o ponteiro aponta: **int y = *x;**
"&"	Endereço que uma variável está guardada na memória: &y;

Importante!

- De modo geral, um ponteiro só pode receber o endereço de memória de uma variável do mesmo tipo do ponteiro.

Praticando...

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

main()
{
    int *p, *p1, x=10;
    float y= 20.0;

    p=&x; // atribuição do endereço de x ao ponteiro *p

    printf("O conteudo apontado por p e %d \n\n", *p);
    printf("O endereco apontado por p e %d \n\n", &p);

    p1=p; // atribuição do ponteiro a outro ponteiro

    printf("O conteudo apontado por p1 e %d \n\n", *p1);

    p=&y;

    printf("O conteudo apontado por p e %d \n\n", *p);
    printf("O conteudo apontado por p e %f \n\n", *p);

    system("pause");
}
```


Não veio nesta aula?

Procure compreender o Código desenvolvido e qualquer dúvida comente com o seu professor!

Referências Bibliográficas

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. *Lógica de Programação*.Ed. Pearson Brasil, 2000.

MANZANO, José Augusto Navarro Garcia; OLIVEIRA, Jair Figueiredo de. *Algoritmos*: Lógica para o desenvolvimento de programação. São Paulo: Érica, 2004.

MIZRAHI, Victorine Viviane. *Treinamento em linguagem C*. São Paulo: McGraw-Hill, 1990.

PEREIRA, Silvio do Lago. Algoritmos e Lógica de Programação em C: Uma abordagem Didática. São Paulo: Érica, 2010.

SANTOS, R. A. R. *Notas de Aula: Arquivo pessoal*. São Paulo: [s.e.], 2018.

SCHILDT, Hebert. *C – Completo e total*. São Paulo: Makron Books, 1995.

Complemente o seu conhecimento

- Acesse o AVA e complemente o seu estudo com as aulas:

"Definição e aplicação de Ponteiros em C",

"Ponteiros para Ponteiros",

"Conceitos sobre aplicação de Ponteiros em Estruturas em C", e

"Ponteiros em Vetores".

- Exercite o seu conhecimento no site:

"https://studio.code.org/s/20-hour"

- Extra - Lista de Exercícios:

"https://goo.gl/9hCXMF"

Desafio Geral:

Faça os exercícios a seguir na linguagem de programação C. Lembre-se que para fazer os exercícios a abstração é de suma importância!

Desafio Geral:

1- Resolver todos os exercícios da lista de exercícios; (Lista de Exercícios: https://goo.gl/9hCXMF);

2- Passar por todas as fases do studio.org e conseguir o certificado final (https://studio.code.org/s/20-hour);

3- Resolver todos os exercícios propostos em cada tópico de aula de uma maneira diferente do que foi apresentada na correção.

4- Desafio extra: Passar por todos os desafios anteriores e realizar o desafio de matrizes (Procure o professor para saber mais detalhes caso tenha atingido esta meta).