UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Sistemas de Informação

Linguagem de Programação I

Professor : Carlos Helano

Objetivo

- Desenvolver a capacidade lógica para construção de algoritmos na resolução de problemas.
- Apresentar uma visão geral do processo de programação.
- Apresentar os recursos da linguagem de programação C.

Conteúdo

- Introdução a Linguagem C (Conceitos básicos)
  - Tipos de Dados;
  - Comandos de Entrada e Saída de Dados;
- Estruturas de Controle
  - Estruturas sequenciais;
  - Estruturas condicionais;
  - Estruturas de repetição;
- Estrutura de Dados Compostas
  - Vetores, Matrizes, Registros (Struct)
- Modularização / Funções
- Ponteiros

Ferramentas para Prática

- Utilizaremos as ferramentas abaixo para o desenvolvimento das práticas
  - Falcon C++ . Disponível : https://sourceforge.net/projects/falconcpp/
  - Repl.it. Disponível : https://repl.it/

Vetores

- Vetores são alocados de maneira contígua na memória.
- São utilizados para “agrupar” diversos dados de um mesmo tipo.
- C não faz nenhuma verificação de limites do vetores. Isto quer dizer que o programador pode avançar o espaço de memória que foi alocado.
- O programador é responsável por manter o acesso ao vetor dentro do limite que foi criado.

Vetores

- Declaração

  tipo nome[tamanho];

- Em C, o índice do vetor começa de 0.
- tipo: Pode ser qualquer tipo padrão ou tipo definido por uma struct.
- nome: O nome que será utilizado para fazer referência ao vetor.
- Tamanho: Varia de 0 até tamanho-1

Vetores

- Declaração

  tipo nome[tamanho];

- O espaço que um vetor ocupa na memória é calculado da seguinte maneira:

  tamanho * (bytes ocupados por cada tipo)

- O operador **sizeof** retorna (em bytes) o tamanho de uma variável ou tipo. Sintaxe:

  – sizeof nome_variavel

  – sizeof (nome_tipo)

Atividades

1- Elabore um programa que leia um vetor de 10 elementos que representam 10 valores de notas (Notas com valor de 0 a 10). Calcule a média das notas e informe quantas notas estão acima da média.

2 - Leia um vetor de 10 elementos inteiros e a partir da leitura de um valor inteiro qualquer verifique se este número pertence ao vetor.

3 - Leia 2 vetores inteiros de 5 elementos e verifique se os 2 vetores contêm os mesmos elementos.

4 - Leia um vetor com n elementos inteiros, e organize os elementos em uma sequência invertida.

Matriz

- Vetores são chamados de Matrizes Unidimensionais.
- Uma Matriz é uma estrutura com 2 ou mais dimensões, capaz de armazenar dados de um mesmo tipo.
- A declaração de uma Matriz bidimensional é parecida com a declaração de um Vetor.

```
tipo_da_variavel   nome_da_matriz [num_linhas][num_colunas];
```

- Exemplos :

```
int   matInt [6][2];
float matReal [5][7];
```

Matriz

- No exemplo abaixo temos uma Matriz ou Vetor Bidimensional

```
#include <stdio.h>
#include <stdlib.h>
main(){
int matriz[3][3];
int cont = 10;
for (int l = 0; l < 3 ; l++)
  for (int c = 0; c < 3 ; c++)
  {
     matriz[l][c] = cont;
     cont = cont + 10;
  }

for (int l = 0; l < 3 ; l++)
{
  for (int c = 0; c < 3 ; c++)
  {
     printf("%d ",matriz[l][c]);
  }
  printf("\n");
}

system("pause");

}
```

Preenchendo a Matriz

Exibindo a Matriz

```
10 20 30
40 50 60
70 80 90
Pressione qualquer tecla para continuar. . .
```

Matriz

- Como exibir os elementos da Diagonal Principal da Matriz

```
printf("\n \n Diagonal Principal \n \n");

for (int l = 0; l < 3 ; l++)
{
  for (int c = 0; c < 3 ; c++)
  {
      if (l == c)
         printf("%d ",matriz[l][c]);
  }
}
```

10	20	30
40	50	60
70	80	90

Registro

- Um registro é uma estrutura de dados heterogênea, formada por uma coleção de variáveis que podem assumir tipos diferentes de dados, inclusive os tipos compostos (vetores, matrizes e registros).

```
struct tipo_endereco {
     char rua[50];
     char bairro[20];
     char cep[10];
  };


tipo_endereco endereco;
```

Criação da estrutura “Registro”

Criação de uma variável do tipo “Registro”

Registro

- As variáveis são agrupadas sob um único nome para a conveniência de manipulação.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  struct tipo_endereco {
     char rua[50];
     char bairro[20];
     char cep[10];
  };
  char nome[50];
  tipo_endereco endereco;

  printf("Nome : ");   gets(nome);
  printf("Rua : ");    gets(endereco.rua);
  printf("Bairro : "); gets(endereco.bairro);
  printf("CEP : ");    gets(endereco.cep);

  printf("\n %s %s %s \n", nome, endereco.rua, endereco.bairro);
  system("pause");
}
```

Vetor de Registros

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  struct tipo_aluno {
     int   matricula;      char  nome[50];
     int   teste;          int   prova;       float media;
  };
  tipo_aluno alunos[3];
```

```
  for (int i = 0; i < 3 ; i++)
  {
    printf("Matricula : "); scanf("%d", &alunos[i].matricula);
    fflush(stdin);
    printf("Nome : ");       gets(alunos[i].nome);
    printf("Teste : ");      scanf("%d", &alunos[i].teste);
    printf("Prova : ");      scanf("%d", &alunos[i].prova);
    alunos[i].media = (alunos[i].teste + alunos[i].prova)/2;
  }
}
```

Ponteiros

- Um **ponteiro** é uma variável cujo conteúdo é o endereço de uma posição de memória.

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

main()
{
  int x;
  int *p;

  x = 8;
  p = &x;
  printf("%d %d \n", x, *p);

  *p = 9;
  printf("%d %d \n", x, *p);

  x = 7;
  printf("%d %d \n", x, *p);

  system("pause");
}
```