Introdução à linguagem C
Aula 05 - Ponteiros

Professor: Racyus Delano
E-mail: racyuspacifico@univicosa.com.br

Ponteiros

- **Variável de ponteiro**
  - Uma variável que armazena um endereço de memória;
    - Permite programas C simular chamadas por referência;
    - Permite o programador criar e manipular estruturas de dados dinâmicas;
  - Precisa ser definido antes de poder ser usado;
    - Deve ser inicializado como NULL ou endereço válido.

Ponteiros

- **Declarando ponteiros**
  - **Formato:**

    <tipo> *variável

    <tipo> *variável = valor inicial

  - **Exemplos:**

```
int *x_ptr;     // Not initialized
double *aPtr = NULL, *bPtr = NULL;
char *grade = NULL;
```

Ponteiros

- **Valores ponteiros**
  - Uma variável de ponteiro tem dois valores associados:
    - **Valor direto**
      - É o endereço de outra célula de memória;
      - Referenciado usando o nome da variável.
    - **Valor indireto**
      - É o valor da célula de memória cujo o endereço é o valor direto do ponteiro;
      - Referenciado usando o operador indireto *.

Ponteiros

- **Operadores ponteiro**
  - Vêm antes do nome da variável:
    - **Operador ***
      - Operador indireto ou operador de desreferenciar;
      - Retorna um sinônimo, alias, ou apelido para o qual seu operando aponta.
    - **Operador &**
      - Endereço do operador;
      - Retorna o endereço de seu operando.

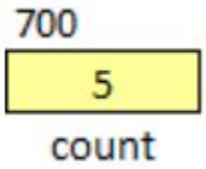
Ponteiros

- **Variáveis de ponteiro**
  - Uma maneira de armazenar um valor em um ponteiro de variável é usar o operador &.
    - O endereço de cont é armazenado em countPtr;
    - Nós dizemos, countPtr aponta para count.

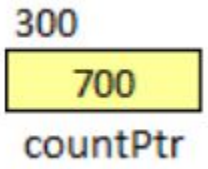
```
int count = 5;
int *countPtr = &count;
```

Ponteiros

- **Variáveis de ponteiro**
  - Suponha que count será armazenado na memória na posição 700 e countPtr será armazenado na memória na posição 300.
  - ```
    int count = 5;
    ```

    - 5 é armazenado em count.
  - ```
    int *countPtr = &count;
    ```

    - O endereço de count é armazenado em countPTR.
  - **<u>Representação gráfica</u>**

Ponteiros

- **Variáveis de ponteiro**
  - Operador *
  - `*countPtr = 10;`
    - Armazena o valor 10 no endereço apontado pelo countPtr.
  - **<u>Representação gráfica</u>**

Ponteiros

- **Variáveis de ponteiro**
  - O caracter * é usado de duas formas:

1. **<u>Declarar que uma variável é um ponteiro</u>**
   a. Uma variável com * em uma declaração significa que a variável será um ponteiro para o tipo indicado ao invés de uma variável regular.
2. **<u>Acessar a posição apontada pelo ponteiro</u>**
   a. Uma variável com * em uma expressão indica o valor em uma posição apontada por um endereço armazenado na variável.

Ponteiros

- **Passando ponteiros por referência na função**
  - Na função usa o operador * nos parâmetros

```
void increment(int *n) {
    *n += 1;  // or (*n)++;
}
```

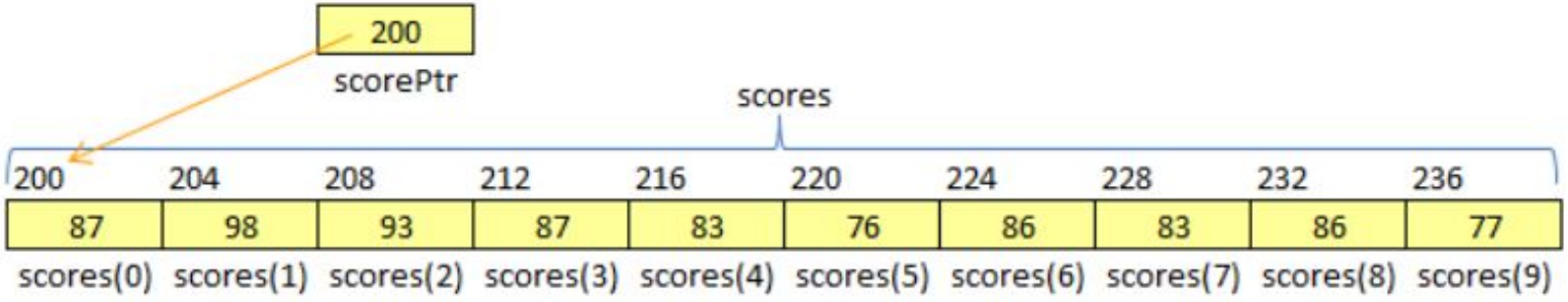
  - Ao chamar a função usa o operador & nos parâmetros

```
int count = 0;
increment(&count);
printf("%d\n", count); // Prints 1
```

Ponteiros

- **Variáveis de ponteiros e Vetores**
  - O nome de um vetor e o endereço do primeiro elemento no vetor representam a mesma coisa.
  - Consequentemente, podemos acessá-los por ponteiros:

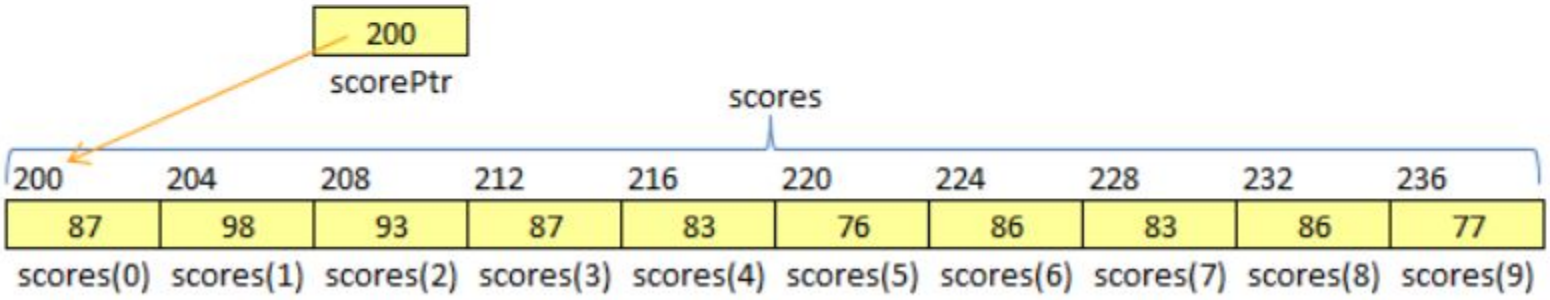
```
int scores[10] = {87,98,93,87,83,76,86,83,86,77};
...
int *scorePtr = NULL;
...
scorePtr = &scores[0];  // Points to first element
```



Ponteiros

- **Variáveis de ponteiros e Vetores**
  - O nome de um vetor é uma constante de ponteiro para o primeiro elemento do vetor.
  - Então, podemos acessá-los como abaixo:

```
int scores[10] = {87,98,93,87,83,76,86,83,86,77};
...
int *scorePtr = NULL;
...
scorePtr = scores;    // Points to array
```



Ponteiros

- **Aritmética de ponteiros e Vetores**
  - Se scorePtr estiver apontando para um elemento específico no vetor e n é um número inteiro, **scorePtr + n** é o valor do ponteiro a n elementos de distância.
  - Podemos acessar os elementos do vetor usando a notação de vetor ou notação de ponteiro;
    - Se **scorePtr** aponta para o primeiro elemento, os duas expressões abaixo são equivalentes:
      - **score[n]** (Notação de vetor)
      - ***(scorePtr + n)** (Notação de ponteiro)

Ponteiros

- **Alocação dinâmica de memória**
  - **Malloc**
    - Aloca dinamicamente um espaço de memória;
    - Alocação contígua, sem inicialização;
    - Precisa instanciar **#include<stdlib.h>;**
    - Usa um argumento
      - Quantidade total de memória requerida
    - Retorna
      - Ponteiro Void em caso de sucesso.
      - NULL em caso de insucesso.

Ponteiros

- **Alocação dinâmica de memória**
  - **Malloc**
    - **Exemplo 1: String**

```
const int str_len = 500;
char *str_ptr = NULL;
...
str_ptr = (char *) malloc(str_len);
if (str_ptr == NULL) {
   printf("Halting: Unable to allocate string.\n");
   exit(1);
}
```

Ponteiros

- **Alocação dinâmica de memória**
  - **Malloc**
    - **Exemplo 2: Inteiro**

```
const int arraySize = 1000;
int *arrayPtr = NULL;
...
arrayPtr = (int *) malloc(arraySize * sizeof(int));
if (arrayPtr == NULL) {
   printf("Halting: Unable to allocate array.\n");
   exit(1);
}
```

Ponteiros

- **Alocação dinâmica de memória**
  - **Free**
    - Libera espaço de memória alocado;
    - Desalocação contígua;
    - Precisa instanciar **#include<stdlib.h>;**
    - Usa um argumento
      - Ponteiro para o início da memória alocada.

Ponteiros

- **Alocação dinâmica de memória**
  - **Free**
    - **Exemplo**

```
const int arraySize = 1000;
int *arrayPtr = NULL;
...
arrayPtr = (int *) malloc(arraySize * sizeof(int));
if (arrayPtr == NULL) {
   printf("Halting: Unable to allocate array.\n");
   exit(1);
}
...
free(arrayPtr);
arrayPtr = NULL;
```

Exercícios sala de aula

1. Desenvolva um programa em C que declare duas variáveis do tipo inteiro e duas do tipo ponteiro de inteiro apontando para essas variáveis. Utilizando ponteiros, o programa deve ler dois números para essas variáveis e os imprimir, realizando as quatro operações básicas de matemática com esses números.

2. Desenvolva um programa em C que declare três variáveis do tipo inteiro e três do tipo ponteiro de inteiro apontando para essas variáveis. Utilizando ponteiros, leia três números e os imprima em ordem crescente. O programa deve apresentar também o endereço de memória desses números.

Exercícios da lista 01

- **Data de entrega: 12/03/2024.**

1. Desenvolva um programa em C que leia seis números e armazene-os em um vetor. Esse programa deve conter ponteiros para manusear o vetor e imprimir os seus valores. O programa deve apresentar também o endereço de memória desses números.

2. Desenvolva um programa em C que leia quatro números e armazene-os em um vetor. Esse programa deve conter ponteiros para manusear o vetor e imprimir os seus valores. Esse programa deve conter ponteiros também para apresentar o maior e o menor número do vetor.