PONTEIROS

Quando uma variável é criada em um programa, um endereço de memória é associado a esta variável e, quando armazenamos um valor dentro da variável, na verdade, estamos armazenando o valor no endereço de memória que está associado a variável utilizada.

O endereço de memória é representado por um valor definido na notação hexadecimal, que é um sistema de numeração em base 16, que utiliza números e letras (de A a F) para representar qualquer valor numérico.

A Figura 1 e a Tabela 1 ilustram como um valor é armazenado na memória do computador.

```
int numero = 10;
int idade = 31;
```

Figura 1 - Criação de variáveis com atribuição de valores

Memória Principal		
Variável	**Endereço**	**Valor**
numero	0x12DC4B3A	10
idade	0x12DC4B3E	31

Tabela 1 - Ilustração da memória do computador

Ponteiros são variáveis que armazenam endereços de memória como valor. Podemos dizer que os ponteiros são variáveis que apontam para o endereço de outra variável.

A utilização de ponteiros permite que programadores possam manipular informações diretamente nos endereços de memória, o que ajuda a melhorar o desempenho do programa, porém tem que ser utilizado com muito cuidado pois a manipulação incorreta pode gerar efeitos colaterais como perda ou corrupção de informações importantes para o programa.

Para declarar um ponteiro, utiliza-se o caractere * antes do nome da variável e para armazenar um endereço dentro do ponteiro basta utilizar o caractere ***&*** antes da variável desejada.

A Figura 2 e a Tabela 2 ilustram o funcionamento de um ponteiro.

```
int numero = 10;
int *ponteiro = &numero;
```

Figura 2 - Criação de um ponteiro

Memória Principal		
Variável	**Endereço**	**Valor**
numero	0x12DC4B3A	10
ponteiro	0x12DC4B3E	0x12DC4B3A

Tabela 2 - Ilustração da memória do computador

Ao utilizar um ponteiro, é possível acessar o endereço de memória que ele armazena e também é possível acessar o valor que está armazenado no endereço de memória guardado no ponteiro. Para acessar o valor de um endereço armazenado no ponteiro também se utiliza o caractere *.

A Figura 3 mostra como acessar as informações através de um ponteiro.

```
int numero = 10;
int *ponteiro = &numero;

cout << "Endereço armazenado no ponteiro: " << ponteiro;
cout << "Valor do endereço armazenado no ponteiro: " << *ponteiro;
```

Figura 3 - Acessando valores pelo ponteiro

Para concluir, podemos manipular o valor de uma determinada variável através de um ponteiro. A Figura 4 mostra como alterar o valor da variável número através do ponteiro que aponta para o seu endereço de memória.

```
int numero = 10;
int *ponteiro = &numero;

*ponteiro = 20;

//Será impresso 20
cout << "Valor da variável 'numero': " << numero;
```

Figura 4 - Manipulação através de ponteiros

REFERÊNCIAS

ASCENCIO, A. F. G.; CAMPOS, E. A. V. Fundamentos da Programação de Computadores. 3. ed. São Paulo: Pearson, 2012.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. Lógica de Programação: A Construção de Algoritmos e Estruturas de Dados. 3. ed. São Paulo: Pearson, 2005.

C++ Pointers, W3Schools, 2020. Disponível em

<https://www.w3schools.com/cpp/cpp_pointers.asp>. Acesso em: 17 de novembro de 2020.