Estrutura de Dados 1

Ponteiros

Profª Juliana Franciscani

Roteiro

01 O que é ponteiro

02 Exemplos

03 Funções e ponteiro

04 Exemplos e exercícios

Ponteiro

- **O que é?**

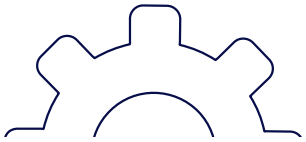
  É uma variável especial que armazena endereços de memória de outra variável. Permite acessar esse valor de forma indireta.

- **Para que serve?**

  ❏ Modificar parâmetros de funções (passagem por referência)

  ❏ Alocar e desalocar memória dinamicamente do sistema

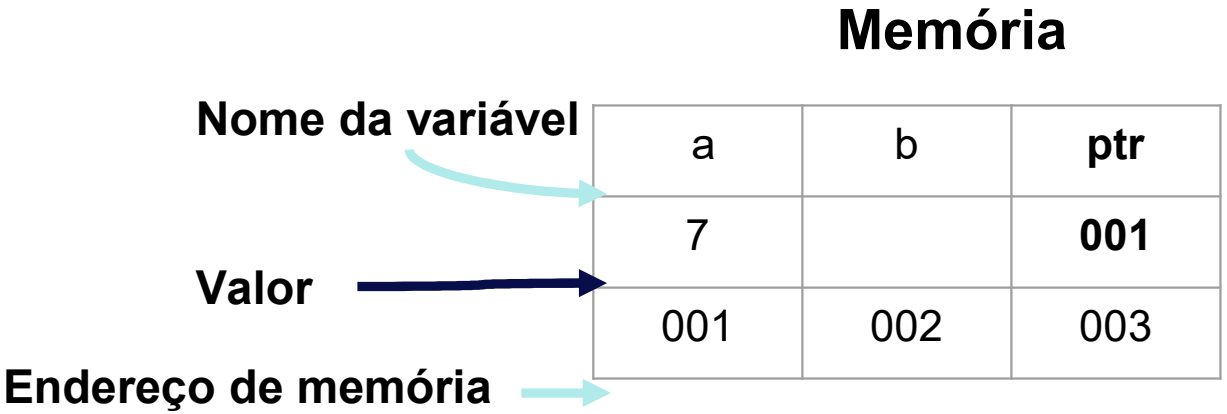
  ❏ Criar estruturas de dados complexas



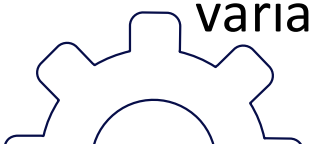
Ponteiro

- **Acesso a memória**

```
int a = 7, b;
int *ptr;
ptr = &a;
```



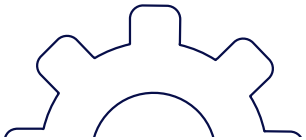
- **Utilização dois operadores:**
  - ❑ O operador & (“endereço de”): resulta no endereço de memória da variável
  - ❑ O operador * (“conteúdo de”): é aplicado a ponteiros, acessa o valor da variável que o ponteiro aponta



Ponteiro

- **Como utiliza em C++?**

```
int x=2, y=10;
int *ptr1, *ptr2;
ptr1 = &x;
ptr2 = nullptr;
cout <<“Valor de ptr1: "<< ptr1;
cout <<“\nValor de x: "<< *ptr1 ;
cout <<“\nValor de x: "<< x ;
ptr2=&y;
*prt2=5;
```



→ Sempre que possível inicialize ponteiros durante sua declaração.

→ Para acessar o valor da variável que o ponteiro aponta usa-se *;

→ Para atribuir um valor a um ponteiro, deve-se usar *nome do ponteiro;

→ Para inicializar um ponteiro use & e o nome da variável (exceto para arrays);

A partir dos códigos abaixo...

Qual é a saída desse programa?

```
int main(){
 int i = 0;
 cout << i << ", ";
 int *ptr = &i;
 *ptr = 1;
 cout << i << ", ";
 i = 10;
 cout << *ptr << endl;
 return 0;
}
```

Quais as possibilidades para exibir o endereço de i?

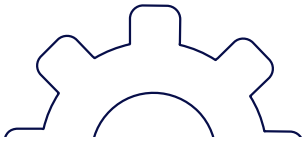
Como altero e exibo o caracter 'P' alterando o ptr?

```
int main(){
 char v = 'I';
 cout << v;
 char *ptr = &v;
 *ptr = 'F';
 cout << v;
 v = 'S';
 cout << *ptr;
...
 return 0;
}
```

Exemplos e Exercícios

A partir dos dados do exemplo anterior, crie o programa completo e verifique qual seria a saída para os valores de y. Exibir o valor e o endereço das variáveis direta e também o valor e o endereço do ponteiro e o valor da variável que ele aponta.

```
int x=2, y=10, *ptr1, *ptr2;
char letra = ‘T’, *ptrLetra;
ptr1 = &x;
ptr2 = nullptr;
cout <<“Valor de ptr: "<< ptr1;
cout <<“\nValor de x de forma indireta: "<< *ptr1 ;
cout << “\nEndereço de ptr1 :” << ....;
cout <<“\nValor de x: "<< x ;
cout <<“\nEndereço de x: "<< ...;
```



```
ptr2=&y;
*prt2=5;
```

Ponteiros e String

```
int main() {
    string nome = "Estrutura";

    string *nomePtr = &nome;

    *nomePtr += " de Dados 1";
    cout << "Nome: "<< nome;
 }
```

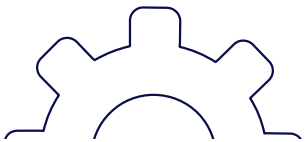
```
 int main() {
    char nome[100];
    char *nPtr = nome;

    strcpy(nome,"Aula");
    cout << nome << endl;

    strcat(nome," Ponteiro");
    cout << nome << endl;
    cout << nPtr << endl;
    cout << *nPtr << endl;
}
```

Ponteiro e Vetor

```
int v[4] = {1, 3, 7, 2}, i;

i=0;
for (i=0; i<5; i++)
   cout<< v[i] << “ “;
```

```
int v[4] = {1, 3, 7, 2}, i;
int  *pv;

pv = v; // ou pv=&v[0];

for (i=0; i<5; i++){
   cout << *pv << “ ”;
   pv++;
}
```

```
int v[4] = {1, 3, 7, 2}, i;
int  *pv;

pv = v; // ou pv=&v[0];

for (i=0; i<5; i++)
   cout << pv[i] << “ ”;
```

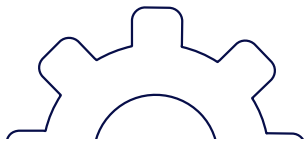


Ponteiros e structs

É possível criar um ponteiro que aponta para uma struct, permitindo acesso indireto às informações.

Há duas formas de acessar às informações:

- Usando o `*` e o ponto: `(*ptr_struct).variável`
- Usando a `->` : `ptr_struct->variável`



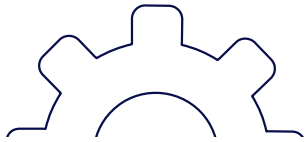
```
struct Pessoa{
    string nome;
    int idade;
};
int main() {
     Pessoa cadPessoa;
     cadPessoa.nome = “Pedro";
     cadPessoa.idade = 33;

     cout << "Nome: " << cadPessoa.nome << endl;
     cout << "Idade: "<< cadPessoa.idade <<endl;

     Pessoa *ptr_pessoa = &cadPpessoa;

     cout << "Nome (via ponteiro): " << ptr_pessoa->nome << endl;
     cout << "Idade (via ponteiro): " << ptr_pessoa->idade << endl;

     ptr_pessoa->idade = 31;
     cout << "Nova idade (via struct): " << cadPessoa.idade << endl;
     cout << "Nova idade (via ponteiro *): " << (*prt_pessoa).idade << endl;
     cout << "Nova idade (via ponteiro ->): " << prt_pessoa->idade << endl;

     return 0;
}
```

Ponteiros e parâmetros de funções

Há duas formas de passagem de parâmetros para funções:

- por valor: os valores dos argumentos da chamada da função são **copiados** para os parâmetros da função

- por referência: os valores dos argumentos da chamada da função são **vinculados** aos parâmetros da função



Ponteiros e Função

```
int main(){
    int x = 10, y = 3;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    trocaValor(&x, &y);
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    return 0;
}
void trocaValor(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

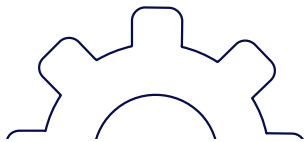
Qual a diferença?

```
int main(){
    int x = 10, y = 3;
    cout << "x = " << x;
    cout << "y = " << y;
    troca(x, y);
    cout << "x = " << x;
    cout << "y = " << y;
    return 0;
}
void troca(int a,int b){
    int temp = a;
    a = b;
    b = temp;
}
```

Função, Ponteiro e Struct

Passar structs por referência:

- Ao passar um ponteiro para uma struct para uma função, a função pode modificar a struct original, evitando a criação de cópias.



```
#include <iostream>
#include <string>
using namespace std;

struct Pessoa {
    string nome;
    int idade;
};

void atualizaIdade(Pessoa *ptrCad);

int main() {
    Pessoa cadPessoa;
    cadPessoa.nome = “Pedro";
    cadPessoa.idade = 30;

    cout << “Idade atual: " << cadPessoa.idade << endl;
    atualizaIdade(&cadPessoa);
    cout << “Nova idade " << cadPessoa.idade << " anos.\n";

    return 0;
}

void atualizaIdade(Pessoa *ptrCad){
       ptrCad->idade +=1;
}
```

Ponteiros e Vetores

Existe uma associação forte entre vetores e ponteiros

Ao declarar **int vetor[5];**

Passar um vetor para uma função consiste em passar o endereço da primeira posição do vetor.

```
imprimeVetor(int vetor[0]);
imprimeVetor(int vetor[]);
imprimeVetor(int *vetor);
```

Exercícios

1. Programa que recebe dois números calcula a soma e exibe o resultado. Utilize ponteiro para manipular pelo menos uma das variáveis e exiba o valor através do ponteiro.
2. Programa que recebe nome, nota válida (0.0 e 100.0) e nome da disciplina. Faça a verificação se o aluno passou ou não. Sabe-se que nota >=60 ele está aprovado e caso contrário está reprovado. Utilize a seguinte estrutura:

```
double nota, *ptrNota;
string nome, disc, *ptrNome;
prtNota = &nota;
ptrNome = &nome;
```

3. Programa que recebe dois números calcula a soma e **exibe o resultado na main()**. Utilize procedimento (não deverá retornar o resultado) para calcular a soma dos números.

4. Programa que recebe informações de um animal (nome, espécie, raça, idade) e armazene em uma struct (Animal). Utilize ponteiro e função/procediemento para receber os dados e depois para exibir os dados. Siga o seguinte padrão

```
void  cadastrarAnimal(Animal *cadAnimal);
void exibirCadastro(Animal *cadAnimal);
```

5. Idem do exercício 2, porém usando strutc e função. Programa que recebe nome, nota válida (0.0 e 100.0) e nome da disciplina. Faça a verificação se o aluno passou ou não. Sabe-se que nota >=60 ele está aprovado e caso contrário está reprovado.

6. Programa que recebe 10 números e os armazena em um vetor, exiba as informações contidas no vetor. Utilize função para armazenar e outra para exibir as informações. O vetor deverá ser passado para as funções usando ponteiro.

Referências

EDELWEISS, N,; GALANTE, R.. Estruturas de dados. Porto Alegre: Bookman, 2009.

SZWARCFITER, J. L.; MARKENZON, L. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro: LTC, 2010.

ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C. 3. ed. rev. e ampl. São Paulo: Cengage Learning, 2011.

Aulas e vídeo aulas do professor André Backes: https://www.facom.ufu.br/~backes/