Laboratório de Programação II

Ponteiros e Alocação Dinâmica de Memória

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

- Declarar variáveis ponteiros
- Manipular ponteiros (operadores & e $*$)
- Funções e ponteiros (passagem por referência)
- Alocação dinâmica de memória
- Liberação de memória

Revisão

- Ponteiros

```
int x = 10, y = 20;

int *pt;      // declaracao
pt = &x;      // inicializacao: pt aponta para x

cout << *pt; // imprime conteudo apontado
             // pelo ponteiro pt

*pt = 66;     // altera o conteudo
              // apontado por pt

pt = &y;      // pt aponta para y;
```

- Atenção: não foi feita alocação dinâmica de memória nesse programa!

Revisão

- Alocação dinâmica de memória

```
int *pt = new int;
// ...
*pt = *pt + 1;
// ...
delete pt;

int *pt_x = new int[100];
// ...
for(int i = 0; i < 100; i++)
    pt_x[i] = i*i;
// ...
delete [] pt_x;
```

Exercícios

Ponteiros

1. Faça um **programa** que declare uma variável `pt` do tipo ponteiro para inteiro e imprima o seu endereço de memória. Em seguida, realize as seguintes operações:
   - Crie uma variável inteira `x`.
   - Leia um valor para `x`.
   - Faça com que `pt` aponte para `x`.
   - Imprima o conteúdo da variável `pt`.
   - Imprima o endereço de `x`.
   - Usando apenas o ponteiro `pt` multiplique `x` por 10 e altere o seu valor.
   - Imprima o conteúdo apontado por `pt`.
   - Some 10 à variável `pt`.
   - Imprima o seu conteúdo novamente.
   - Qual é a saída? O que significa?

Exercícios

Ponteiros

2. Implemente a função `troca` que troca o conteúdo de duas variáveis inteiras `a` e `b`. Faça um **programa** que teste a função implementada. Protótipo:

```
void troca(int *a, int *b);
```

3. Dados dois números inteiros `num` e `div`, faça uma função para calcular e retornar o quociente e o resto da divisão inteira de `num` por `div`. Considerar o seguinte protótipo:

```
void divisao(int num, int div, int *q, int *r);
```

onde:

- `num` é dividendo;
- `div` é o divisor;
- `q` é o quociente;
- `r` é o resto.

Exercícios

Ponteiros

4. Implemente uma única função que receba um vetor de números inteiros (`vet`) e o seu tamanho (`tam`) e:
   - conte o total de elementos pares;
   - conte o total de elementos impares;
   - conte o total de elementos negativos;
   - e por fim, retorne **verdadeiro** se existirem números negativos no vetor, ou retorne **falso**, caso contrário.

   Considere o seguinte protótipo:

```
bool func(int tam, int vet[], int *par,
          int *impar, int *negativos);
```

Exercícios

Alocação Dinâmica

5. Faça um **programa** que leia um número inteiro `n` e aloque um vetor com `n` inteiros de forma dinâmica (use o operador `new`). Em seguida o programa deve **calcular a média** dos elementos desse vetor e, por fim, deve desalocar (use o operador `delete`) a memória usada para armazenar os seus elementos.

Exercícios

Alocação Dinâmica

6. Implemente uma função que calcule o produto escalar entre dois vetores do tipo de dados `float`. No programa principal você deve ler o tamanho `n` dos vetores, os quais devem ser alocados dinamicamente usando `new`. Depois, você deve ler os dados dos vetores e chamar a função para calcular o produto escalar. Por fim, use o operador `delete` para desalocar toda memória alocada de forma dinâmica. Protótipo:

```
float prodEscalar(int n, float x[], float y[]);
```

Ex. O produto escalar entre $\mathbf{x} = [1, 2, 3]$ e $\mathbf{y} = [4, 5, 6]$ é dado por:

$$\mathbf{x} \cdot \mathbf{y} = 1 \times 4 + 2 \times 5 + 3 \times 6 = 32.$$