Estrutura de Dados e Algoritmos

Funções

Procedimentos e Funções

- Procedimentos e Funções: Subconjuntos de códigos (comandos);
- Diferenças:
  - Procedimentos: não retorna nenhum valor. Consequentemente, não tem tipo;
  - Funções: retornam algum tipo de valor. Para tanto, devem ter o tipo do valor que retorna;
- Ambos podem ou não ter parâmetros;

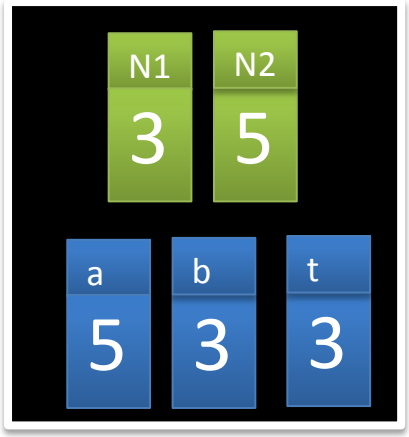
Procedimentos e Funções

- Exemplos:
  - void ImprimeOla();
  - void ImprimeQtdeXPreco(int qtde, float preco);
  - float QuadradoDePI();
  - Long int SomaNumeros(int *VetNum, int Tamanho);

Passagem de Parâmetros

- Passagem de parâmetros por valor:
  - A função recebe uma cópia da variável que é fornecida quando é invocada. Todas as alterações feitas dentro da função não vão afetar as variáveis globais:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  troca(N1,N2);
}
```

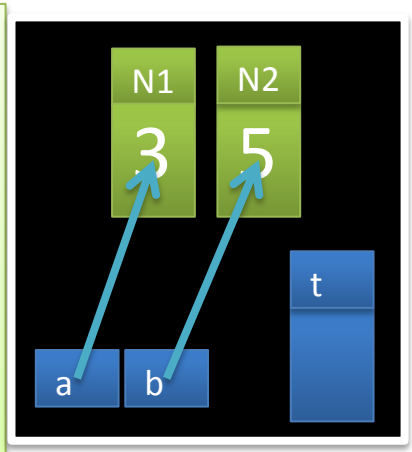



```
void troca(int a, int b)
{
   int t;
   t=a;
   a=b;
   b=t;
}
```

Passagem de Parâmetros

- Passagem de parâmetros por referência:
  - A função recebe uma referência da variável que é fornecida quando é invocada. Todas as alterações feitas dentro da função vão afetar os variáveis globais:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  troca(&N1,&N2);
}
```




```
void troca(int *a, int *b)
{
   int t;
   t=*a;
   *a=*b;
   *b=t;
}
```

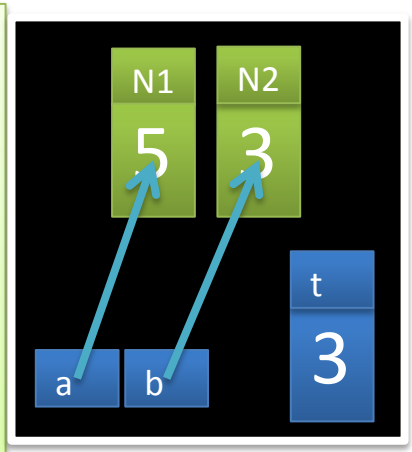
Passagem de Parâmetros

- Implementação em C:
  - Parâmetros por valor: OK;
  - Parâmetros por referência: NÃO;
- Implementação em C++:
  - Parâmetros por valor: OK;
  - Parâmetros por referência: OK;
- Alternativa para se implementar parâmetros por referência em linguagem C: Usar ponteiros.

Passagem de Parâmetros

- Passagem de parâmetros por referência:
  - A função recebe uma referência da variável que é fornecida quando é invocada. Todas as alterações feitas dentro da função vão afetar os variáveis globais:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  troca(&N1,&N2);
}
```



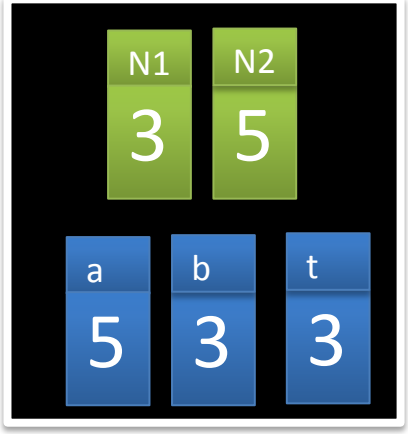

```
void troca(int *a, int *b)
{
   int t;
   t=*a;
   *a=*b;
   *b=t;
}
```

Passagem de Parâmetros

- Passagem de parâmetros por valor em C/C++:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  printf("Antes: %d, %d\n",N1,N2);
  troca(N1,N2);
  printf("Depois: %d, %d\n",N1,N2);
}
```

```
void troca(int a, int b)
{
   int t;
   t=a;
   a=b;
   b=t;
}
```



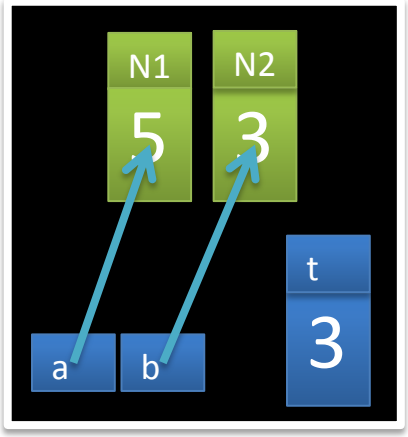

```
Antes: 3, 5
Depois: 3,5
```

Passagem de Parâmetros

- Passagem de parâmetros por referência em C:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  printf("Antes: %d, %d\n",N1,N2);
  troca(&N1, &N2);
  printf("Depois: %d, %d\n",N1,N2);
}
```

```
void troca(int *a, int *b)
{
   int t;
   t=*a;
   *a=*b;
   *b=t;
}
```



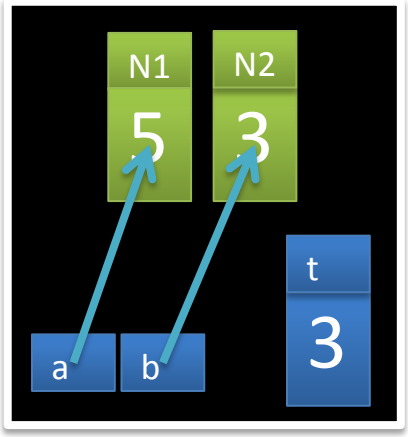

```
Antes: 3, 5
Depois: 5,3
```

Passagem de Parâmetros

- Passagem de parâmetros por referência em C++:

```
#include<stdio.h>
main()
{
  int N1=3, N2=5;
  printf("Antes: %d, %d\n",N1,N2);
  troca(N1, N2);
  printf("Depois: %d, %d\n",N1,N2);
}
```

```
void troca(int &a, int &b)
{
   int t;
   t=a;
   a=b;
   b=t;
}
```




```
Antes: 3, 5
Depois: 5,3
```

Prática...

```
#include<stdio.h>
main()
{
  int N1=3, N2=5, N3=1;
  printf("Antes: %d, %d\n",N1,N2,N3);
  ordena(&N1,&N2,&N3);
  printf("Depois: %d, %d\n",N1,N2,N3);
}
```

Escreva a função ordena em Linguagem C de forma a fazer com que o programa principal imprima as mensagens abaixo:

```
Antes: 3, 5, 1
Depois: 1, 3, 5
```

```
#include<stdio.h>
main()
{
  int N1=3, N2=5, N3=1;
  printf("Antes: %d, %d\n",N1,N2,N3);
  ordena(N1,N2,N3);
  printf("Depois: %d, %d\n",N1,N2,N3);
}
```

Escreva a função ordena em Linguagem C++ de forma a fazer com que o programa principal imprima as mensagens abaixo:

```
Antes: 3, 5, 1
Depois: 1, 3, 5
```

Ponteiros para Funções

- As linguagens C/C++ permitem o acesso ao endereço de memória de funções → Ponteiros para Funções;
- Declaração de um ponteiro para função:

  TipoRetorno (*nomeponteiro)(<lista de parâmetros>);

- Um ponteiro para função é definido pelo tipo de retorno e por sua lista de parâmetros:

Ponteiros para Funções

- Em C/C++, o nome de uma função é um ponteiro.


```
int f(int);          //declara a função f
int (*pf)(int);      //declara o ponteiro de função pf
pf = &f;             //atribui o endereço de f a pf
```

```
int f(int n)
{
  ....;
  return ...;
}
```

Ponteiros para Funções

- O ponteiro para função pode ser usado para fazer uma chamada a função, com a mesma sintaxe de chamadas comuns.

```
void * (*ptr) (size t, size t);
ptr = calloc;
...
int *p = (int *) ptr(2, 8);
```

- Ponteiros para função permitem a construção de programas dinâmicos e podem aumentar a organização do código.

Ponteiros para Funções

```
#include<stdio.h>
#include “exemplo.c”
main()
{
  float R, R2;
  float (*p) (int i, int j);
  p = exemplo;
  R   = p(12,45);
  R2 = (*p)(12,45);
  printf(“R= %f, R2 = %f”, R, R2);
}
```

```
float exemplo(int n1, int n2)
{
  return 3.14159*n1*n2;
}
```

Ponteiros para Funções

- Parâmetros do tipo ponteiro de função.

```
#include<stdio.h>
#include “funcoes.c”
main()
{
  float psq, psc;
  psq = AplicaF(5,quadrado);
  psc = AplicaF(5,cubo);
  printf(“Soma dos quadrados: %f”,psq);
  printf(“Soma dos cubos: %f”,psc);
}
```

```
float quadrado(int pnum)
{
  return pnum*pnum;
}
float cubo(int pnum)
{
  return pnum*pnum*pnum;
}

float AplicaF(int n, float (*f) (int k))
{
  int ptermo;
  float psoma = 0;
  for(ptermo=1; ptermo<=n; ptermo++)
    psoma += f(ptermo);
  return psoma;
}
```

Prática...

Construa um programa que receba da linha de comando dois inteiros e a descrição da operação que será efetuada sobre eles (soma, subtracao, multiplicacao, divisao ou resto). O programa deve possuir uma função para cada uma das operações mencionadas (ambas devem receber dois números inteiros e retornar um valor inteiro), devendo se utilizar do conceito de "ponteiro para função" na seleção da função adequada a ser executada.

Vetores de Ponteiros para Funções

- Vetores de ponteiros para funções permite a implementação de um conjunto de diferentes funções que podem ser selecionadas por índices.
- Sintaxe geral:

TipoRetorno (*PtrFunc[Qtde])(<lista de parâmetros>);

Vetores de Ponteiros para Funções

- Exemplos:

```
void (*series[10]) (unsigned int pq);
```

```
double (*calculos[3]) (double *pvalores, unsigned int n);
```

```
char*  (*formato[5]) (char *frase, unsigned int n);
```

Vetores de Ponteiros para Funções

- func → nome de um objeto;
- func[] → é um vetor;
- (*func[]) → é um vetor de ponteiros;
- (*func[])() → é um vetor de ponteiros para funções;
- int (*func[])() → é um vetor de ponteiros para funções;

Vetores de Ponteiros para Funções

- Pode-se atribuir o endereço de funções já existentes ao vetor:

```
int  func1(int i, int j);
int  func2(int i, int j);
int  (*func[])(int, int) = {func1, func2};
```

Prática...

1. Complete a função main() para que o programa VetPontFuncoes funcione corretamente.

   Obs: não altere as outras funções do programa.

Menu Principal

```
########################## ESTATISTICAS DE SERIES ################

##########################          MENU          ################


[G] - Gerar nova Serie
[T] - Todas Estatisticas
[S] - Soma
[M] - Media
[D] - Desvio Padrao
[P] - Soma de Potencias
[Q] - Soma dos Quadrados
[Qualquer Outra Tecla] - Encerrar
Digite a Opcao Desejada:
```

Gerar Nova Série

```
########################          SERIE (20 Termos)          ################
1o. Termo:      1.000000
2o. Termo:      0.763153
3o. Termo:      0.272831
4o. Termo:      0.274389
5o. Termo:      0.206450
6o. Termo:      0.711743
7o. Termo:      0.616849
8o. Termo:      0.836646
9o. Termo:      0.015816
10o. Termo:     0.305794
11o. Termo:     0.621687
12o. Termo:     0.938166
13o. Termo:     0.354573
14o. Termo:     0.770297
15o. Termo:     0.516205
16o. Termo:     0.418713
17o. Termo:     0.656469
18o. Termo:     0.027962
19o. Termo:     0.301669
20o. Termo:     0.407898

################################################################################
```

Todas Estatísticas

```
########################## ESTATISTICAS ################
Soma:    9.889945

Media:   0.494497

DesvioPadrao:    0.005675

SomaPotencias:   4.269876

SomaQuadrados:   7.160536

################################################################
Pressione qualquer tecla para continuar. . .
```

Prática...

2. Construa um programa que implemente vetores de ponteiro para funções para calcular a área de figuras geométricas: triângulo retângulo, triângulo isósceles, quadrado, trapézio e círculo;
3. Construa um programa que implemente vetores de ponteiro para funções que formate uma cadeia de caracteres, conforme as seguintes opções: 1. Coloque todas letras minúsculas, 2. Coloque todas letras maiúsculas, 3. Coloque a primeira letra maiúscula e o restante minúsculas, 4. Coloque todas palavras começando com letra maiúscula e o restante em letras minúsculas;