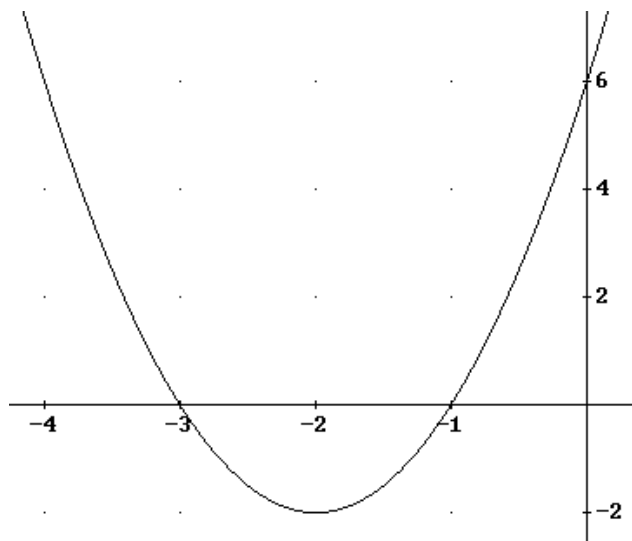


# Funções

# Funções - Matemáticas

- Função quadrática  $y = ax^2 + bx + c$ 
  - Entrada:  $x$
  - Saída:  $y$



# Função em programação

- Funções são blocos de código que podem ser nomeados e chamados de dentro de um programa.
  - **printf()**: função que escreve na tela
  - **scanf()**: função que lê o teclado

# Função

- Facilitam a estruturação e reutilização do código.
  - Estruturação: programas grandes e complexos são construídos bloco a bloco.
  - Reutilização: evitam a cópia desnecessária de trechos de código que realizam a mesma tarefa
  - Permite abstração

# Função

- Em linguagens imperativas, TODOS os programas usam funções
- No C, o programa SEMPRE começa executando a função **main**.

```
#include <stdio.h>

int main(void)
{
    printf("Olá, Mundo!");
    return 0;
}
```

# Função - Estrutura

- Forma geral de uma função:

```
tipo_retornado nome_função(parâmetros){  
    conjunto de declarações e comandos  
}
```

# Função - Exemplo

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

# Função - Exemplo

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Nome da  
função**






# Função - Exemplo

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Parâmetro de entrada da  
função**



# Função - Exemplo

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Tipo de saída da  
função**



# Função - Exemplo

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x)) ;  
3. }
```

**código da  
função**




# Função - Parâmetros

- A declaração de parâmetros é uma lista de variáveis juntamente com seus tipos:
  - tipo nome1, tipo nome2, ... , tipoN nomeN

```
01 //Declaração CORRETA de parâmetros
02 int soma(int x, int y){
03     return x + y;
04 }
05
06 //Declaração ERRADA de parâmetros
07 int soma(int x, y){
08     return x + y;
09 }
```

# Função - Parâmetros

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }  
4.  
5. int main() {  
6.     double entrada = 10.0;  
7.     double saida = logistica(entrada);  
8.     printf("%f",  , saida);  
9.     return 0;  
10. }
```

**Variável saida recebe o valor da função**

# Função - Main

- A função main é especial:
  - É a primeira a ser chamada no programa
    - Todo programa tem um!
  - Seu retorno pode indicar se o programa executou corretamente (retorno **0**) ou não (retorno **!= 0**)
  - Seus parâmetros, quando existem, são os parâmetros passados para o programa quando foi executado (**int argc, char \*argv[]**)

# Função sem retorno

- Funções sem retorno (ou **procedimentos**) devem ter o tipo de retorno **void**
- Exemplo: função para imprimir mensagem de boas-vindas do programa

```
1. void saudacao() {  
2.     printf("Ola usuario! Digite o comando que quer  
   executar, ou ? para ajuda.");  
3. }  
4. int main() {  
5.     saudacao();  
6.     ...  
7.     return 0;  
8. }
```

# Função sem retorno

- Porque o importante pode ser a ação **colateral** da função, e não o seu valor de saída:
  - Impressão de uma mensagem
  - Ligar/desligar um componente do hardware
  - ...
- Porque a função **sempre executa sem erro**:
  - `void exit();`



# Comando return

- Uma função pode ter mais de uma declaração **return**.

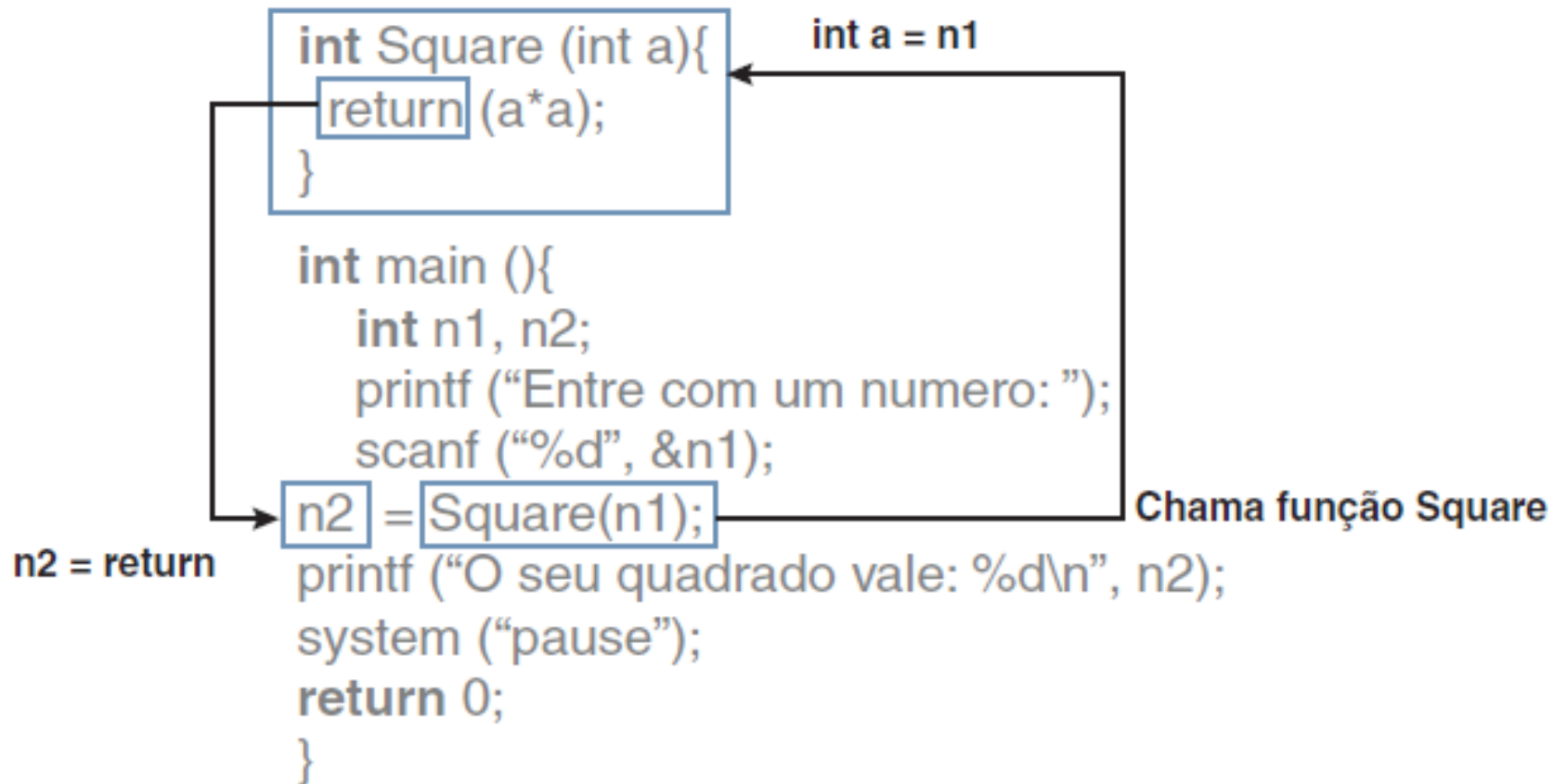
```
01  int maior(int x, int y){  
02      if(x > y)  
03          return x;  
04      else  
05          return y;  
06  }
```

- Quando o comando **return** é executado, a função termina imediatamente.

# Exemplo

```
01  #include <stdio.h>
02  #include <stdlib.h>
03
04  int Square (int a){
05      return (a*a);
06  }
07
08  int main(){
09      int n1,n2;
10      printf("Entre com um numero: ");
11      scanf("%d", &n1);
12      n2 = Square(n1);
13      printf("O seu quadrado vale: %d\n", n2);
14      system("pause");
15      return 0;
16  }
```

# Exemplo



# Declaração de Funções

- Funções devem ser definidas ou declaradas antes de serem utilizadas, ou seja, antes da cláusula main.
- A definição (protótipo) apenas indica a existência da função:

**tipo\_retornado nome\_função(parâmetros);**

- Desse modo ela pode ser declarada após a cláusula main().

# Exemplo

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  //protótipo da função
04  int Square (int a);
05
06  int main(){
07      int n1,n2;
08      printf("Entre com um numero: ");
09      scanf("%d", &n1);
10      n2 = Square(n1);
11      printf("O seu quadrado vale: %d\n", n2);
12      system("pause");
13      return 0;
14  }
15
16  int Square (int a){
17      return (a*a);
18  }
```

# Escopo de variáveis

- Escopo
  - Define onde e quando a variável pode ser usada.
- Escopo global
  - Fora de qualquer definição de função
  - Tempo de vida é o tempo de execução do programa
- Escopo local
  - Bloco ou função

# Escopo de variáveis

```
#include <stdio.h>
#include <stdlib.h>
void func1(){
    int x;//variável local
}
void func2(){
    int x;//variável local
}
int main(){
    int x;
    scanf("%d",&x);
    if(x == 5){
        int y=1;
        printf("%d\n",y);
    }
    system("pause");
    return 0;
}
```

- Escopo global
- Escopo local
- Escopo local dentro de outro escopo local

# Funções: escopo de variáveis

- `int teste(int x) {`
- `...`
- `}`
  
- `int main() {`
- `int y;`
- `for(int i=0;i<10;i++) {`
- `if(i < 5) {`
- `int a;`
- `} else {`
- `int b;`
- `}`
- `}`
- `}`



# Funções: escopo de variáveis

- `int teste(int x) {`
- `...`
- `}`

Escopo de x

- `int main() {`
- `int y;`
- `for(int i=0;i<10;i++) {`
- `if(i < 5) {`
- `int a;`
- `} else {`
- `int b;`
- `}`
- `}`
- `}`

# Funções: escopo de variáveis

- `int teste(int x) {`
- `...`
- `}`

Escopo de y

- `int main() {`
- `int y;`
- `for(int i=0;i<10;i++) {`
- `if(i < 5) {`
- `int a;`
- `} else {`
- `int b;`
- `}`
- `}`
- `}`

# Funções: escopo de variáveis

```
• int teste(int x) {  
• ...  
• }  
  
• int main() {  
• int y;  
•   for(int i=0;i<10;i++) {  
•       if(i < 5) {  
•           int a;  
•       } else {  
•           int b;  
•       }  
•   }  
• }
```

Escopo de i

# Funções: escopo de variáveis

```
• int teste(int x) {  
• ...  
• }  
  
• int main() {  
• int y;  
•     for(int i=0;i<10;i++) {  
•         if(i < 5) {  
•             int a;  
•         } else {  
•             int b;  
•         }  
•     }  
• }
```

Escopo de a

# Funções: escopo de variáveis

```
• int teste(int x) {  
• ...  
• }  
  
• int main() {  
• int y;  
•   for(int i=0;i<10;i++) {  
•       if(i < 5) {  
•           int a;  
•       } else {  
•           int b;  
•       }  
•   }  
• }
```

Escopo de b

# Funções e escopo

- As variáveis locais (variáveis de uma função) são armazenadas em um modelo de pilha:
  - Cada nova variável criada é adicionada ao topo da pilha
  - Ao terminar o bloco, eliminamos todas as variáveis daquele bloco da pilha
- O mesmo vale para chamada de funções

# Funções e escopo

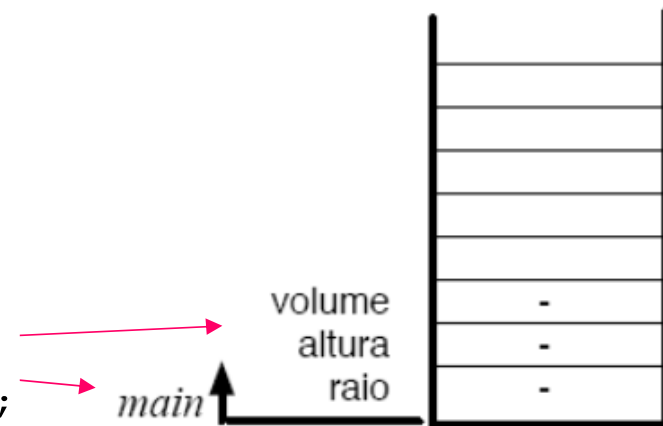
```
#include <stdio.h>
```

```
#define PI 3.14159
```

```
float volume_cilindro(float r, float h) {  
    float v;  
    v = PI * r * r * h;  
    return v;  
}
```

```
int main(void) {  
    float raio, altura, volume;  
    printf("Entre com o valor do raio: ");  
    scanf("%f", &raio);  
    printf("Entre com o valor da altura: ");  
    scanf("%f", &altura);  
  
    volume = volume_cilindro(raio, altura);  
  
    printf("Volume do cilindro = ");  
    printf("%f", volume);  
  
    return 0;  
}
```

**float raio, altura, volume;**



# Funções e escopo

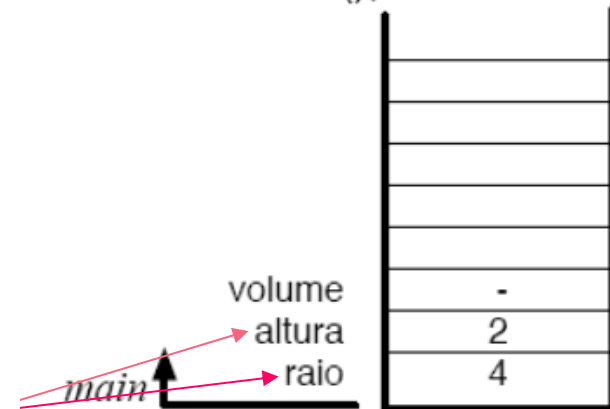
```
#include <stdio.h>
```

```
#define PI 3.14159
```

```
float volume_cilindro(float r, float h) {  
    float v;  
    v = PI * r * r * h;  
    return v;  
}
```

```
int main(void) {  
    float raio, altura, volume;  
    printf("Entre com o valor do raio: ");  
    scanf("%f", &raio);  
    printf("Entre com o valor da altura: ");  
    scanf("%f", &altura);  
  
    volume = volume_cilindro(raio, altura);  
  
    printf("Volume do cilindro = ");  
    printf("%f", volume);  
  
    return 0;  
}
```

```
raio = readfloat();  
altura = readfloat();
```





# Funções e escopo

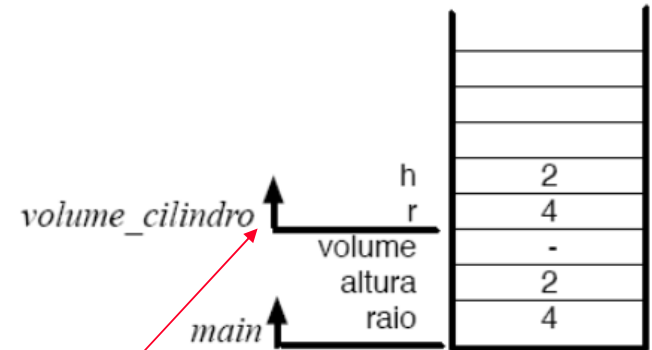
```
#include <stdio.h>
```

```
#define PI 3.14159
```

```
float volume_cilindro(float r, float h) {  
    float v;  
    v = PI * r * r * h;  
    return v;  
}
```

```
int main(void) {  
    float raio, altura, volume;  
    printf("Entre com o valor do raio: ");  
    scanf("%f", &raio);  
    printf("Entre com o valor da altura: ");  
    scanf("%f", &altura);  
  
    volume = volume_cilindro(raio, altura);  
  
    printf("Volume do cilindro = ");  
    printf("%f", volume);  
  
    return 0;  
}
```

volume = volume\_cilindro(raio, altura)



# Funções e escopo

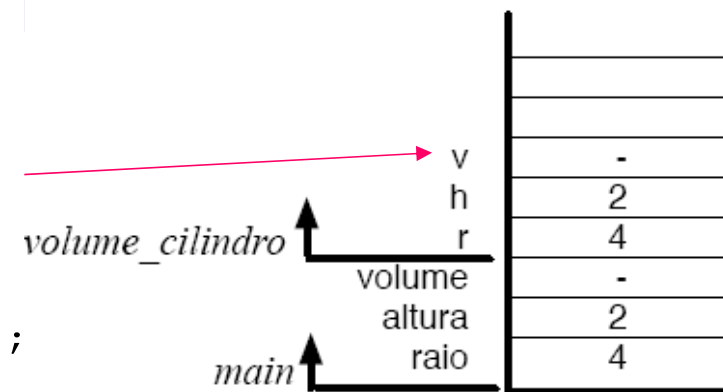
```
#include <stdio.h>
```

```
#define PI 3.14159
```

```
float volume_cilindro(float r, float h) ;  
    float v;  
    v = PI * r * r * h;  
    return v;  
}
```

```
int main(void) {  
    float raio, altura, volume;  
    printf("Entre com o valor do raio: ");  
    scanf("%f", &raio);  
    printf("Entre com o valor da altura: ");  
    scanf("%f", &altura);  
  
    volume = volume_cilindro(raio, altura);  
  
    printf("Volume do cilindro = ");  
    printf("%f", volume);  
  
    return 0;  
}
```

**float v;**



# Funções e escopo

```
#include <stdio.h>
```

```
#define PI 3.14159
```

```
float volume_cilindro(float r, float h) {  
    float v;  
    v = PI * r * r * h;  
    return v;  
}
```

```
int main(void) {  
    float raio, altura, volume;  
    printf("Entre com o valor do raio: ");  
    scanf("%f", &raio);  
    printf("Entre com o valor da altura: ");  
    scanf("%f", &altura);  
  
    volume = volume_cilindro(raio, altura);  
  
    printf("Volume do cilindro = ");  
    printf("%f", volume);  
  
    return 0;  
}
```

volume = volume\_cilindro(raio, altura);

