Faculdade de Computação - FACOM

Bacharelado em Sistemas de Informação

FACOM32201 - Algoritmos e Programação II

Prof. Thiago Pirola Ribeiro

FUNÇÕES - Parte 2

Exemplos

```
1  // função sem retorno e sem parâmetros de entrada
2  void MensagemBoasVindas() {
3      printf("\n");
4      printf("======================\n");
5      printf(" Seja Bem-Vindo \n");
6      printf("======================\n");
7      printf("\n");
8  }
9  int main()
10 {
11     // Chamando a função
12     MensagemBoasVindas();
13     ....
14
```

Exemplos

```
1  // função sem parâmetros de entrada, mas com retorno
2  char MenuPrincipal(){
3      char op;
4      printf("Escolha uma opção: \n\n");
5      printf("1 - Novo Jogo\n");
6      printf("2 - Carregar Jogo\n");
7      printf("3 - Sair\n");
8      setbuf(stdin,NULL);
9      scanf("%c",&op);
10     return op;
11 }
12 int main() {
13     char escolha;
14     MensagemBoasVindas();
15     escolha = MenuPrincipal();
16
```

Exemplos

```
1  // função com retorno e com parâmetros
2  int VerificaAprovacao(double nota, int faltas) {
3      int aprovado = 1;
4      if ( (faltas > 18) || (nota < 60.0) ) {
5          aprovado = 0;
6      }
7      return aprovado;
8  }
9  int    main(){
10     int ap;
11     ap = VerificaAprovacao(4.0, 5);
12     if (ap) {
13         printf("Aprovado!!!");
14     }
15
```

Exemplos

```
1  // supor 0 -> aprovado
2  // supor 1 -> reprovado
3  // supor 2 -> reprovado por falta
4
5  if (VerificaAprovacao(nota, faltas) == 0) {
6     printf("Aprovado!!!");
7  } else if (VerificaAprovacao(nota, faltas) == 1) {
8     printf("Reprovado!!!");
9  } else if (VerificaAprovacao(nota, faltas) == 2) {
10    printf("Reprovado por falta");
11 }
12
```

Exemplos

```
1  // supor 0 -> aprovado
2  // supor 1 -> reprovado
3  // supor 2 -> reprovado por falta
4
5  ap = VerificaAprovacao(nota, faltas);
6
7  if (ap == 0) {
8     printf("Aprovado!!!");
9  } else if (ap == 1) {
10    printf("Reprovado!!!");
11 } else {
12    printf("Reprovado por falta");
13 }
14
```

Exemplos

```
1  // função com retorno e com parâmetros
2  int VerificaAprovacao(double nota, int faltas) {
3      int aprovado = 1;
4      if ( (faltas > 18) || (nota < 60.0) ) {
5          aprovado = 0;
6      }
7      return aprovado;
8  }
9
10 int main(){
11     ...
12     if (VerificaAprovacao(nota, faltas) != 0) {
13         printf("Aprovado!!!");
14     }
15
```

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 escrita após a cláusula **main()**.

Exemplo

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

int Square (int a){
    return (a*a);
}
void main(){
    int n1, n2;
    printf("Entre com um numero: ");
    scanf("%d", &n1);
    n2 = Square(n1);
    printf("O quadrado vale: %d\n", n2);
    system("pause");
}
```

Exemplo – usando protótipo

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int Square (int a); //  protótipo da função
5 int main(){
6    int n1, n2;
7    printf("Entre com um numero: ");
8    scanf("%d", &n1);
9    n2 = Square(n1);
10   printf("O quadrado vale: %d\n", n2);
11   system("pause");
12 }
13
14 int Square (int a){
15   return (a*a);
16 }
```

Escopo de Variáveis

Escopo de variáveis

Escopo

É o conjunto de regras que determinam o uso e a validade de variáveis nas diversas partes do programa.

Variáveis Locais, Variáveis Globais e Parâmetros formais.

Escopo

- **Variáveis locais** são aquelas que só têm validade dentro do bloco no qual são declaradas.

Escopo

- **Variáveis locais** são aquelas que só têm validade dentro do bloco no qual são declaradas.
  - Um bloco começa quando abrimos uma chave e termina quando fechamos a chave.

Escopo

- **Variáveis locais** são aquelas que só têm validade dentro do bloco no qual são declaradas.
  - Um bloco começa quando abrimos uma chave e termina quando fechamos a chave.
  - Ex.: variáveis declaradas dentro da função.

Escopo

- **Variáveis globais** são declaradas fora de todas as funções do programa.

Escopo

- **Variáveis globais** são declaradas fora de todas as funções do programa.
- Elas são conhecidas e podem ser alteradas por todas as funções do programa.

Escopo

- **Variáveis globais** são declaradas fora de todas as funções do programa.
- Elas são conhecidas e podem ser alteradas por todas as funções do programa.
  - Quando uma função tem uma variável local com o mesmo nome de uma variável global a função dará preferência à variável local.

Escopo

- **Variáveis globais** são declaradas fora de todas as funções do programa.
- Elas são conhecidas e podem ser alteradas por todas as funções do programa.
  - Quando uma função tem uma variável local com o mesmo nome de uma variável global a função dará preferência à variável local.

Atenção

Evite usar variáveis globais!!!

Escopo

- **Parâmetros formais** são declarados como sendo as entradas de uma função.

Escopo

- **Parâmetros formais** são declarados como sendo as entradas de uma função.
  - O parâmetro formal é uma variável local da função.

Escopo

- **Parâmetros formais** são declarados como sendo as entradas de uma função.
  - O parâmetro formal é uma variável local da função.
  - Ex.: float square(float x); // x é um parâmetro formal

Escopo de variáveis

```c
int main()
{
    int i;
    i = 2;
    printf("Valor de i (antes do bloco): %d\n", i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Valor de i (dentro do bloco): %d\n", i);
    }
    printf("Valor de i (depois do bloco): %d\n", i);
    return 0;
}

```

Escopo - Mapa de Memória

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5			
6			
7			
8			
9			
10			

Escopo de variáveis

```c
int main()
{
    int i;
    i = 2;
    printf("Valor de i (antes do bloco): %d\n", i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Valor de i (dentro do bloco): %d\n", i);
    }
    printf("Valor de i (depois do bloco): %d\n", i);
    return 0;
}
```

Escopo - Mapa de Memória

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5	lx	i	int
6			
7			
8			
9			
10			

Escopo de variáveis

```c
int main()
{
    int i;
    i = 2;
    printf("Valor de i (antes do bloco): %d\n", i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Valor de i (dentro do bloco): %d\n", i);
    }
    printf("Valor de i (depois do bloco): %d\n", i);
    return 0;
}
```

Escopo - Mapa de Memória

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5	3	i	int
6			
7			
8			
9			
10			

Escopo de variáveis

```c
int main()
{
    int i;
    i = 2;
    printf("Valor de i (antes do bloco): %d\n", i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Valor de i (dentro do bloco): %d\n", i);
    }
    printf("Valor de i (depois do bloco): %d\n", i);
    return 0;
}
```

Escopo - Mapa de Memória

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5			
6			
7			
8			
9			
10			

Escopo de variáveis

```c
int main()
{
    int i;
    i = 2;
    printf("Valor de i (antes do bloco): %d\n", i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Valor de i (dentro do bloco): %d\n", i);
    }
    printf("Valor de i (depois do bloco): %d\n", i);
    return 0;
}
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5			
6			
7			
8			
9			
10			

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5	lx	i	int
6			
7			
8			
9			
10			

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5	3	i	int
6			
7			
8			
9			
10			

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	2	i	int
2			
3			
4			
5			
6			
7			
8			
9			
10			

Endereços reais

```c
int main() {
    // verificando endereços das variáveis
    printf("\n\n Verificando os enderecos \n\n");
    int i;
    i = 2;
    printf("Endereco de i (antes do bloco): %u\n", &i);

    // abertura de um bloco
    {
        int i;
        i = 3;
        printf("Endereco de i (dentro do bloco): %u\n", &i);
    }
    printf("Endereco de i (depois do bloco): %u", &i);
    return 0;
}
```

Endereços reais

```
 Verificando os enderecos

Endereco de i (antes do bloco): 3350425520
Endereco de i (dentro do bloco): 3350425524
Endereco de i (depois do bloco): 3350425520
```

Escopo local em Funções

- Função: declarada na lista de parâmetros da função ou definida dentro da função

```
1       int minha_fun (int x, int y) {
2           int i,j;
3       }
```

Escopo local em Funções

- Função: declarada na lista de parâmetros da função ou definida dentro da função

```
1       int minha_fun (int x, int y) {
2           int i,j;
3       }
```

- Variáveis x, y, i e j são locais a função e não são acessíveis a nenhuma outra função ou ao programa principal.

Escopo local em Funções

- Função: declarada na lista de parâmetros da função ou definida dentro da função

```
1       int minha_fun (int x, int y) {
2           int i,j;
3       }
```

- Variáveis x, y, i e j são locais a função e não são acessíveis a nenhuma outra função ou ao programa principal.
- O valor retornado pode ser armazenado diretamente na variável que recebe o valor na chamada da função ou em um local temporário que depois será copiado.

Exemplo

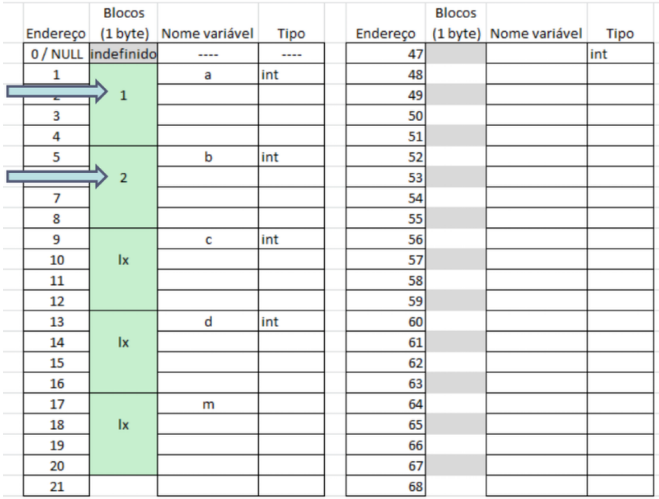
```
1  int maior(int a, int b){
2     if ( a > b)
3        return a;
4     else
5        return b;
6  }
7  int    main(){
8     int a,b,c,d,m;
9  --> Parou Aqui <--
10    a = 1;
11    b = 2;
12    m = maior(a,b);
13    printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
14    c = -1;
15    d = -50;
16    printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17    return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	Ix	a	int
2			
3			
4			
5	Ix	b	int
6			
7			
8			
9	Ix	c	int
10			
11			
12			
13	Ix	d	int
14			
15			
16			
17	Ix	m	
18			
19			
20			
21			
22			
23			
24			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47			int
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			

Exemplo

```
1  int maior(int a, int b){
2      if ( a > b)
3      return a;
4      else
5      return b;
6  }
7  int    main(){
8      int a,b,c,d,m;
9      a = 1;
10     b = 2;
11 --> Parou Aqui <--
12     m = maior(a,b);
13     printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
14     c = -1;
15     d = -50;
16     printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17     return 0;
18 }
```


Endereço
Blocos (1 byte)
Nome variável
Tipo
0 / NULL
indefinido

1
a
int
1
2
3
4
5
b
int
2
7
8
9
c
int
lx
10
11
12
13
d
int
lx
14
15
16
17
m
lx
18
19
20
21
Endereço
Blocos (1 byte)
Nome variável
Tipo
47
int
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

Exemplo

```
1  int maior(int a, int b){
2  --> Parou Aqui <--
3      if ( a > b)
4          return a;
5      else
6          return b;
7  }
8  int    main(){
9      int a,b,c,d,m;
10     a = 1;
11     b = 2;
12     m = maior(a,b);
13     printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
14     c = -1;
15     d = -50;
16     printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17     return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1		a	int
2	1		
3			
4			
5		b	int
6	2		
7			
8			
9		c	int
10	lx		
11			
12			
13		d	int
14	lx		
15			
16			
17		m	
18	lx		
19			
20			
21			
22			
23			
24			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47		a	int
48	1		
49			
50			
51		b	int
52	2		
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			

Exemplo

```
1  int maior(int a, int b){
2     if ( a > b)
3        return a;
4     else
5        return b;
6  }
7  int    main(){
8     int a,b,c,d,m;
9     a = 1;
10    b = 2;
11    m = maior(a,b);
12    printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
13 --> Parou Aqui <--
14    c = -1;
15    d = -50;
16    printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17    return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	1	a	int
2			
3			
4			
5	2	b	int
6			
7			
8			
9	lx	c	int
10			
11			
12			
13	lx	d	int
14			
15			
16			
17	2	m	
18			
19			
20			
21			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			

Exemplo

```
1  int maior(int a, int b){
2     if ( a > b)
3        return a;
4     else
5        return b;
6  }
7  int    main(){
8     int a,b,c,d,m;
9     a = 1;
10    b = 2;
11    m = maior(a,b);
12    printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
13    c = -1;
14    d = -50;
15 --> Parou Aqui <--
16    printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17    return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	1	a	int
2			
3			
4			
5	2	b	int
6			
7			
8			
9	-1	c	int
10			
11			
12			
13	-50	d	int
14			
15			
16			
17	2	m	
18			
19			
20			
21			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			

Exemplo

```
1  int maior(int a, int b){
2  --> Parou Aqui <--
3      if ( a > b)
4          return a;
5      else
6          return b;
7  }
8  int    main(){
9      int a,b,c,d,m;
10     a = 1;
11     b = 2;
12     m = maior(a,b);
13     printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
14     c = -1;
15     d = -50;
16     printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
17     return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	1	a	int
2			
3			
4			
5	2	b	int
6			
7			
8			
9	-1	c	int
10			
11			
12			
13	-50	d	int
14			
15			
16			
17	2	m	
18			
19			
20			
21			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47	-1	a	int
48			
49			
50			
51	-50	b	int
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			

Exemplo

```
1  int maior(int a, int b){
2     if ( a > b)
3        return a;
4     else
5        return b;
6  }
7  int    main(){
8     int a,b,c,d,m;
9     a = 1;
10    b = 2;
11    m = maior(a,b);
12    printf("\n0 maior valor entre (%d,%d) eh %d", a,b,m);
13    c = -1;
14    d = -50;
15    printf("\n0 maior valor entre (%d,%d) eh %d", c,d,maior(c,d));
16 --> Parou Aqui <--
17    return 0;
18 }
```

Endereço	Blocos (1 byte)	Nome variável	Tipo
0 / NULL	indefinido	----	----
1	1	a	int
2			
3			
4			
5	2	b	int
6			
7			
8			
9	-1	c	int
10			
11			
12			
13	-50	d	int
14			
15			
16			
17	2	m	
18			
19			
20			
21			

Endereço	Blocos (1 byte)	Nome variável	Tipo
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			

Faculdade de Computação - FACOM

Bacharelado em Sistemas de Informação

Prof. Thiago Pirola Ribeiro