

Programação de Computadores IV

Subprogramação

Slides cedidos por Bruno Augusto Dorta Marques

Subprogramas

- Pedacos de código que são chamadas pelo programa principal ou por outros subprogramas
- Vantagens
 - Defeitos podem ser facilmente corrigidos, pois as
 - modificações são feitas em um único local
 - Defeitos podem ser evitados, pois a replicação de código pode levar a modificações incompletas
 - O código fica mais legível
 - Facilita o design de algoritmos, pois um problema é dividido em subproblemas

Programa de exemplo

```
int main (){  
  
    float a, b, c;  
  
    scanf("%f%f", &a, &b);  
    c = a * b / 2.0;  
    printf("a vale %f \n", a);  
    printf("b vale %f \n", b);  
    printf("a * b / 2.0 vale: %f \n", c);  
  
    a = a / 2.0;  
    b = b / 2.0;  
    c = a * b / 2.0;  
    printf("a vale %f \n", a);  
    printf("b vale %f \n", b);  
    printf("a * b / 2.0 vale: %f \n", c);  
}
```

}

- É muito comum que vários trechos de código se repitam ao longo de um programa.

}

Utilizando Subprogramas

```
int main (){

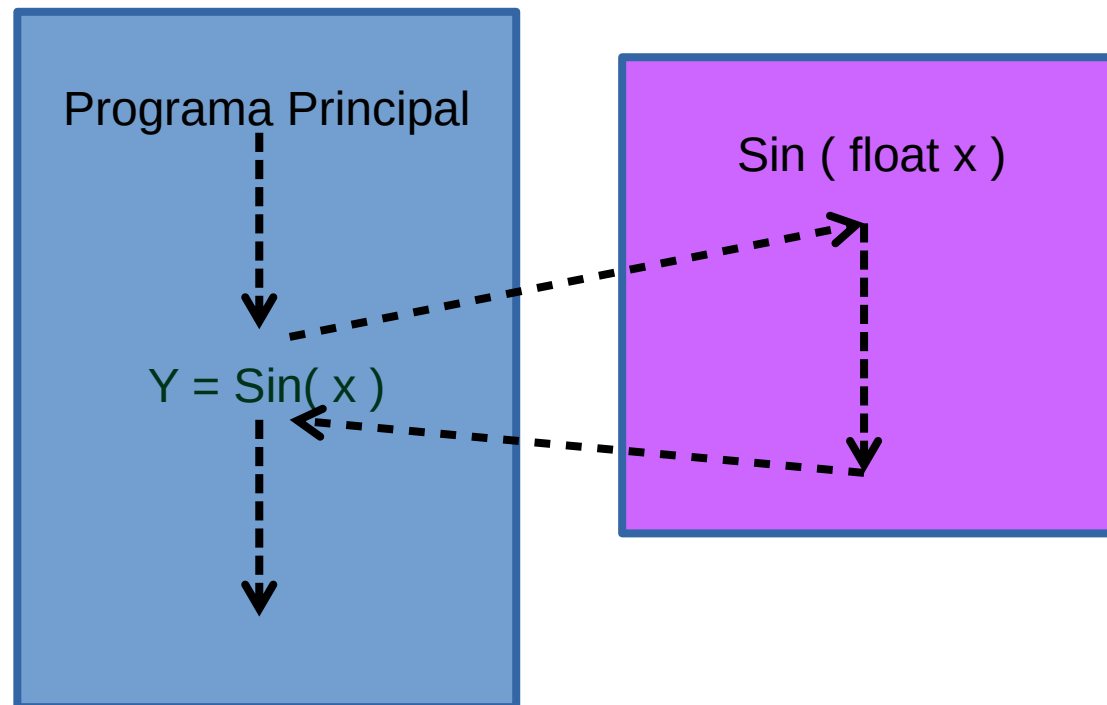
    float a, b, c;
    scanf("%f%f", &a, &b);

    mostra(a, b);
    a = a / 2.0;
    b = b / 2.0;
    mostra(a, b);
}

void mostra(float a, float b){
    float c;
    c = a * b / 2.0;
    printf("a vale %f \n", a);
    printf("b vale %f \n", b);
    printf("a * b / 2.0 vale: %f \n", c);
}
```

Fluxo de Controle

- O mesmo acontece quando usamos uma das **funções pré-definidas** do C



Subprogramas

- Função

- Sempre **retorna um valor**
- A chamada é feita pelo identificador (nome) da função e argumentos entre parênteses, separados por vírgula
 - Exemplo: **sin(x)**
- O valor retornado é do tipo especificado na declaração da função

Detalhes importantes

O nome dos argumentos de funções e sub-rotinas **não precisa** ser igual ao nome das variáveis

O C faz o casamento dos nomes

```
troca(a, b);  
void troca(int x, int y){  
    ...  
}
```



- Variáveis declaradas no programa principal **não podem ser acessadas** pelo subprograma, apenas aquelas passadas como argumento
- Variáveis declaradas no subprograma **não podem ser acessadas** pelo programa principal

Função

Sintaxe

```
<tipo_do_retorno> <nome_da_função>(<tipo1><arg1>, <tipo2><arg2>, ...,  
<tipon><argn>){
```

```
    <declaração de variáveis locais>
```

```
    <corpo da função>
```

```
    return <valor_de_retorno ou variável_de_retorno>
```

```
}
```

Exemplo simples de funções

```
void function1(int a){  
    a = 2* a + 5;  
    ...  
}
```

```
void conta(float a, int b){  
    a = a * 3.0 / b;  
    printf("%f", a);  
}
```

```
int func2(int a,int b){  
    a = a+ b;  
    return a;  
}
```

```
char converte (int a){  
    if(a == 1){  
        return 'a';  
    }else{  
        return 'b'; }  
}
```

Funções

- Declaração de variáveis em Funções
 - As variáveis declaradas dentro de uma função pertencem somente a ela (variáveis locais)

Passagem de parâmetro por valor

- O compilador copia o valor da variável em um novo endereço de memória.
- Alterações na nova variável não afeta a variável anterior.
- Por padrão o C utiliza esse tipo de passagem de parâmetro.
- Exemplo de chamada de função:
`function(a, b);`
- Exemplo de declaração de função :
`void function (int a, int b)`

Passagem de parâmetro por valor

```
int main (void){  
    a = 2;  
    func( a, 2);  
    printf("O Valor de a fora da função é %d \n", a);  
}  
  
void func(int a, int b){  
    a = 4  
    printf("O Valor de a dentro da função é %d \n", a);  
}
```

Passagem de parâmetro por referência

- Passagem de argumentos por referência
 - A mesma região de memória do programa principal é utilizada pelos argumentos da função
 - Alterações no valor das variáveis passadas como argumento são refletidas no programa principal
 - Necessário utilizar ponteiros

Passagem de parâmetro por referência

- Exemplo (Declaração de função):

```
void swap(int *x, int *y) {  
    int temp;  
    temp = *x;  
    *x = *y;  
    *y = temp;  
}
```

Passagem de parâmetro por referência

- Exemplo (chamada de função):

```
int main(void){  
    int a = 100;  
    int b = 200;  
    swap(&a, &b);  
    return 0;  
}
```