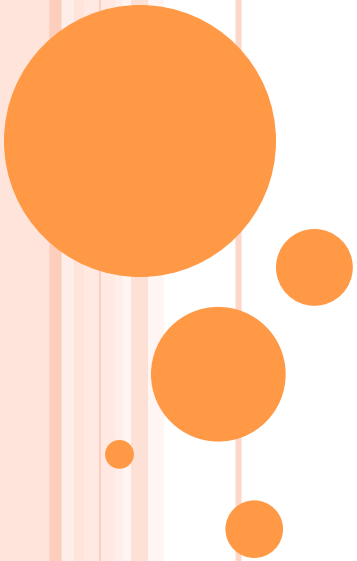

Programação Estruturada

Funções

Modularização de Programas

- Funções
  - Dividem grandes tarefas de computação em tarefas menores → Modularização
  - Além disso, permitem que trechos de código sejam reutilizados
  - Até agora, criamos programas apenas com a função principal → `main()`

Funções

- Em C, a definição de uma função tem a seguinte forma

```
tipo_retorno nome_função (parâmetros) {

      Declarações e comandos

}
```

- Os parâmetros são declarados como:
  - `tipo nome1, tipo nome2, tipo nome3 ...`
- A comunicação entre as funções é feita através da passagem de parâmetros ou através de variáveis globais

Funções

- Em C, não existe “procedimento”, somente função
- Um “procedimento”é uma função que não retorna nada:

```
void escreve_menu() {
        printf(“MENU \n”);
        printf(“1 - somar\n”);
        printf(“2 - subtrair\n”);
        printf(“3 - fim\n”);
}
```

Funções

- Se o tipo da função for omitido, o *default* é o tipo int
- Para que a função retorne um valor para sua função chamadora é usado o comando `return`

```
int soma(int A, int B) {
    int C;
    C = A + B;
    return(C);
}
```

Funções

- Se o tipo da função for omitido, o *default* é o tipo int
- Para que a função retorne um valor para sua função chamadora é usado o comando `return`

```
int soma(int A, int B) {
    int C;
    C = A + B;
    return(C);
}
```

```
int soma(int A, int B) {
    return(A + B);
}
```

Funções

- O tipo da função sempre indica o tipo do valor que será retornado
- Uma função pode possuir mais de um `return`
- A função termina sua execução quando encontrar um `return` ou quando for encontrado o seu fim
- Uma função pode ser declarada antes ou depois da função `main`

Funções - Exemplos

```
int soma(int A, int B) {
    return(A + B);
}

int main() {
    int n1, n2, resultado;
    printf("Digite dois valores inteiros: ");
    scanf("%d %d", &n1, &n2);
    resultado = soma(n1,n2);
    printf("O resultado é: %d", resultado);
    return 0;
}
```

Funções - Exemplos

```
int soma(int A, int B) {
    return(A + B);
}

int main() {
    int n1, n2;
    printf("Digite dois valores inteiros: ");
    scanf("%d %d", &n1, &n2);
    printf("O resultado é: %d", soma(n1,n2));
    return 0;
}
```

Funções - Exemplos

```
int par(int N) {
     if (N % 2 == 0)
         return 1;
     else
         return 0;
}
int main() {
     int N;
     printf("Digite um valor inteiro: ");
     scanf("%d", &N);
     if (par(N))
         printf("O número é PAR");
     else
         printf("O número é ÍMPAR");
     return 0;
}
```

FUNÇÕES - EXEMPLOS

```
char *diaSemana(int i) {
   switch (i) {
      case  1:
            return("DOMINGO");
      case  2:
            return("SEGUNDA");
      case  3:
            return("TERÇA");
      case  4:
            return("QUARTA");
      case  5:
            return("QUINTA");
      case  6:
            return("SEXTA");
      case  7:
            return("SÁBADO");
      default:
           return("Inválido");
   }
}
```

```
double PI() {
     return 3.1416;
}


int soma(int A, int B) {
     return(A + B);
}


void escreve_menu() {
     printf("MENU \n");
     printf("1 - somar\n");
     printf("2 - subtrair\n");
     printf("3 - fim\n");
}
```

Exercícios

1. Faça um programa que leia N valores em dólar e que transforme para real. Deve ser feita uma função que receba o valor em dólar, a cotação e retorne o equivalente em real.

2. Faça um programa imprima na tela um Menu que permita o usuário escolher as seguintes operações: +, -, /, *.
   - Se um operador inválido for digitado deve ser impresso um erro
   - Se o usuário digitar ! o programa deve ser finalizado
   - Se o usuário escolher uma das quatro operações, o programa deve ler dois números reais e efetuá-los
   - Faça funções para escrever o Menu, realizar a operação e verificar se o operador é válido

Escopo de Variáveis

- Relembrando...
  - Variáveis Globais são todas as variáveis declaradas fora de qualquer função e podem ser vistas por todas as funções do programa.
  - Uma variável Local só pode ser reconhecida dentro da função/bloco onde ela foi declarada
  - O escopo de uma variável corresponde a parte do programa onde a variável pode ser usada
    - Existem variáveis com escopo global ou local

Exemplos

```
int A, B, C;

int teste1(int A) {
     int D;
     ...
}

int teste2() {
     int C;
     ...
}

int main() {
     int B;
     A = teste1(B);
     C = teste2();
     return 0;
}
```

Exemplos

```
int A, B, C;

int teste1(int A) {
      int D;
      ...
}

int teste2() {
      int C;
      ...
}

int main() {
      int B;
      A = teste1(B);
      C = teste2();
      return 0;
}
```

A e D são locais

B e C são globais

Exemplos

```
int A, B, C;

int teste1(int A) {
      int D;
      ...
}

int teste2() {
      int C;
      ...
}

int main() {
      int B;
      A = teste1(B);
      C = teste2();
      return 0;
}
```

A e D são locais

B e C são globais

C é local

A e B são globais

Exemplos

```
int A, B, C;

int teste1(int A) {
      int D;
      ...
}

int teste2() {
      int C;
      ...
}

int main() {
      int B;
      A = teste1(B);
      C = teste2();
      return 0;
}
```

A e D são locais
B e C são globais

C é local
A e B são globais

B é local
A e C são globais

Exercício

```
int A, B, C;

int teste(int A) {
      int D = 2;
      A++;
      C += A;
      D -= (A+B);
      return D+1;
}
int main() {
      A = 5;
      B = 6;
      B -= A;
      C = B + A;
      B = B + teste(B);
      printf("A = %d, B = %d, C = %d", A, B, C);
      return 0;
}
```

Qual valor será escrito para A, B e C ao final do programa?