Disciplina:
Programação Computacional*

Prof. Fernando Rodrigues
e-m@il: fernandorodrigues@sobral.ufc.br

Aula 09: Programação em C

- Operadores relacionais
- Estruturas de seleção
- Estruturas de repetição
- Desvio de fluxo

Operadores relacionais

Estabelecem relações entre os dados de um programa.

Operador relacional	Significado	Precedência
==	igual	baixa
!=	diferente	baixa
<	menor	alta
<=	menor ou igual	alta
>	maior	alta
>=	maior ou igual	alta

Operadores relacionais tem precedência mais baixa que os operadores aritméticos.

a é menor do que b

```
a < b
```

a é igual a zero

```
a == 0
```

a é maior do que a soma de b e c

```
a > b + c
```

a é diferente de zero

```
a != 0
```

Verdadeiro ou falso em C

Expressão relacional

Expressão envolvendo dados e/ou operadores aritméticos, e operadores relacionais.

A avaliação de uma expressão relacional produz um valor booleano (verdadeiro ou falso), que é convertido para um valor numérico:

- 0, se a expressão for avaliada como falsa.
- 1, se a expressão for avaliada como verdadeira.

```
5 == 5.0
```

1, pois não há diferença entre variáveis de tipos numéricos.

```
5 == '5'
```

0, pois o valor decimal do caractere '5' é 53.

Verdadeiro ou falso em C

Expressão lógica

Expressão envolvendo dados e/ou operadores aritméticos e/ou operadores relacionais e/ou operadores lógicos.

A avaliação de uma expressão lógica produz um valor booleano (verdadeiro ou falso), que é convertido para um valor numérico:

- 0, se a expressão for avaliada como falsa.
- 1, se a expressão for avaliada como verdadeira.

Generalizando...

- Qualquer valor ou expressão diferente de zero será considerada como verdadeira.
- Qualquer valor ou expressão igual à zero será considerada como falsa.

Estrutura condicional(seleção) simples

if()

Estrutura utilizada para avaliar uma condição simples.

A sintaxe da estrutura `if()` é:

```
if (expressão)
{
  // comandos a serem executados se a expressão for verdadeira.
}
```

Atenção

Não se deve usar ";" após o **if()**.

Bloco de código

As instruções que deverão ser executadas, no caso da `expressão` da estrutura **`if`**`( )` ser avaliada como verdadeira, deverão ser escritas dentro de um bloco delimitado pelos símbolos `{` e `}`.

Dica importante

Para facilitar a visualização, os comandos dentro de um bloco também devem ser recuados.

A omissão dos símbolos `{` e `}` só será permitida se o bloco de instruções se constituir de apenas 1 comando.

No caso da `expressão` da estrutura **`if`**`( )` ser considerada falsa, todo o bloco de comandos será ignorado e o programa prossegue a partir do primeiro comando após o bloco.

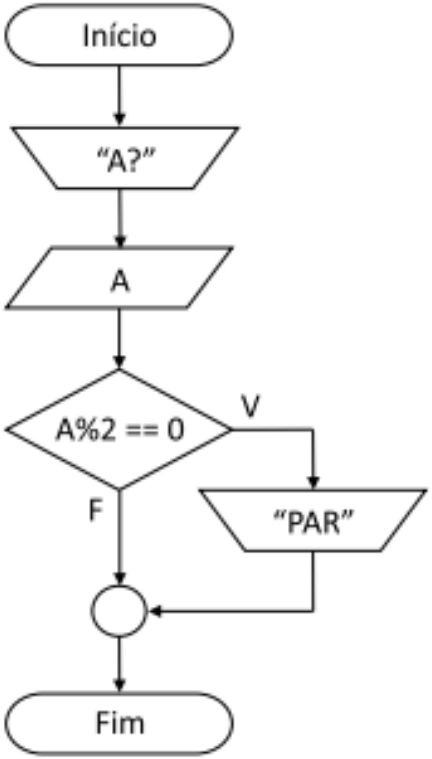
Ex

Considere o seguinte trecho de código:

```
aprovado = 0;
if (media >= 7.0)
{
  aprovado = 1;
  puts("PARABÉNS!!!");
}

if (aprovado == 0)
  puts("Você foi reprovado!");
```

E se os delimitadores de bloco do primeiro **`if`**`( )` forem removidos?

Exemplo: Ler um número inteiro e informar se o mesmo é par.




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

main()
{
  int a;

  printf("Digite um número inteiro: ");
  scanf("%d", &a);
  if (a % 2 == 0)
  {
    printf("O numero digitado é par.\n");
  }
  system("PAUSE");
}
```

Exemplo: Ler 3 valores numéricos distintos e exibir o maior deles.

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

int main()
{
  float a, b, c;
  float maior;

  printf("Digite 3 números: ");
  scanf("%f %f %f", &a, &b, &c);
  if ( a > b && a > c ) { maior = a; }
  if ( b > a && b > c ) { maior = b; }
  if ( c > a && c > b ) { maior = c; }
  printf("O maior numero digitado é %f\n", maior);
  system("PAUSE");
  return(0);
}
```

Exemplo: Verificar se um número real x pertence ao intervalo [–1, 1].

Versão 1: testa se está dentro.

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

int main()
{
  float x;

  printf("Digite um número: ");
  scanf("%f", &x);
  if (x >= -1 && x <= 1)
  {
    printf("Pertence.\n");
  }
  system("PAUSE");
  return(0);
}
```

Estrutura condicional composta (if()-else)

if()-else

Estrutura utilizada para avaliar uma condição composta.

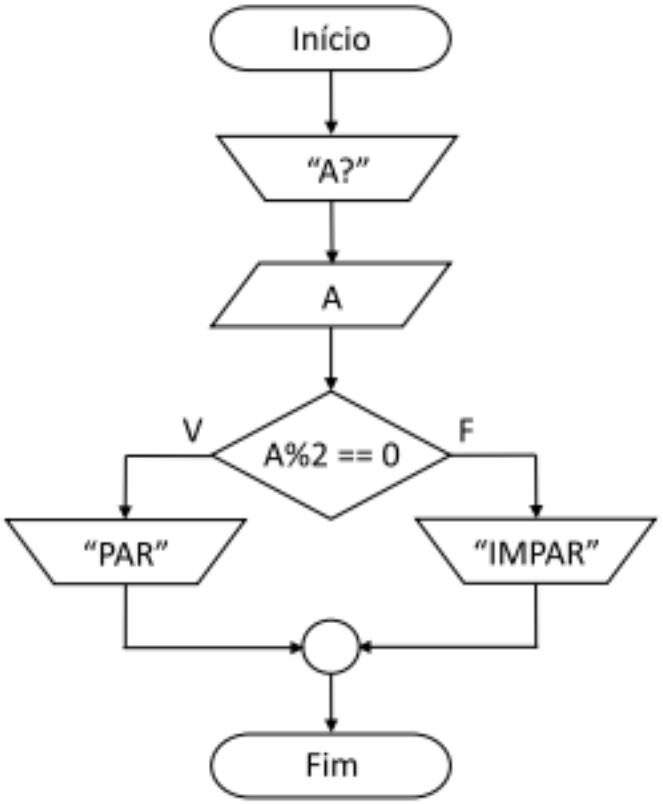
A sintaxe da estrutura `if()-else` é:

```
if (expressão)
{
  // comandos a serem executados se a expressão for verdadeira.
}
else
{
  // comandos a serem executados se a expressão for falsa.
}
```

Atenção

Não se deve usar ";" após o **if()**, nem após o **else**.

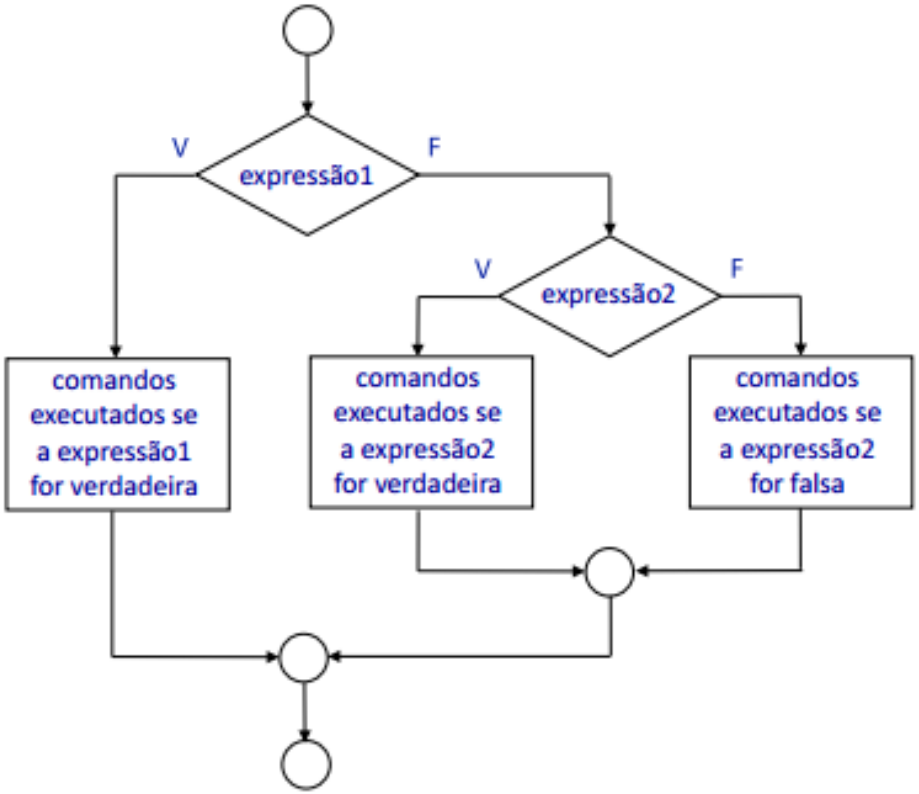
Exemplo: Ler um número inteiro e informar se o mesmo é par ou ímpar.




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

int main()
{
  int a;

  printf("Digite um número inteiro: ");
  scanf("%d", &a);
  if (a % 2 == 0)
  {
    printf("O numero digitado é par.\n");
  }
  else
  {
    printf("O numero digitado é impar.\n");
  }
  system("PAUSE");
  return(0);
}
```

Aninhamento

Técnica de seqüenciar os testes de modo a melhorar o desempenho do programa.



Aninhamento

Se a média M for maior ou igual a 7.0, não há necessidade de realizar outros testes.

```
if ( M >= 7.0 )
  printf("Aprovado.\n");
else
  if ( M >= 5.0 && M < 7.0 )
    printf("Realizar Prova 5.\n");
  else
    printf("Reprovado.\n");
```

Atenção

Uma cláusula **`else`** sempre pertence ao **`if`** imediatamente anterior. Caso isso não for desejável, deve-se utilizar os delimitadores de bloco **{** e **}**.

A estrutura condicional múltipla switch()

O aninhamento de várias estruturas compostas **`if()-else`** (ou **`else-if`**) permite o tratamento de expressões condicionais que permitem mais de duas alternativas.

Se a expressão resultar em um valor do tipo **`int`** ou **`char`**, é possível utilizar a estrutura condicional múltipla **`switch()`**.

```
switch(expressão)
{
  case constante1:
    //comandos1
    break;

  case constante2:
    //comandos2
    break;

  ...

  case constanteN:
    //comandosN
    break;

  default:
    //comandos
}
```

A estrutura condicional múltipla switch()

1. A expressão é avaliada.

2. O fluxo de execução é desviado para o rótulo **`case`** cujo valor equivale ao da expressão. Serão executados todos os comandos até que seja encontrado o comando **`break`**.

3. A execução do comando **`break`** desvia o fluxo de execução para o primeiro comando após a estrutura **`switch()`**.

4. Se nenhum **`case`** corresponde ao valor da expressão, então o fluxo de execução será desviado para o rótulo **`default`** (opcional).

Cuidado!

A omissão do comando **`break`** causará a execução dos **`case`**'s subseqüentes.

A estrutura condicional múltipla switch() - Exemplo:

```c
#include <stdio.h>

main()
{
    float num1, num2, result;
    char oper;
    int valido = 1;
    printf("Digite a operacao a ser executada de forma infixada: (operando real_1 sinal da operacao operando real_2)\n");
    scanf("%f%c%f",&num1,&oper,&num2);
    fflush(stdin);
    switch(oper){
        case '+':
            result = num1+num2;
            break;
        case '-':
            result = num1-num2;
            break;
        case '*':
            result = num1*num2;
            break;
        case '/':
            result = num1/num2;
            break;
        default:
            printf("Operador não definido!\n");
            valido = 0;
    }
    if(valido){
        printf("O resultado de %.2f %c %.2f eh %f\n",num1,oper,num2,result);
    }
}
```

```
Digite a operacao a ser executada de forma infixada: (operando real_1 sinal da operacao operando real_2)
2.5*4
O resultado de 2.50 * 4.00 eh 10.000000
```

Estruturas de repetição

Definição

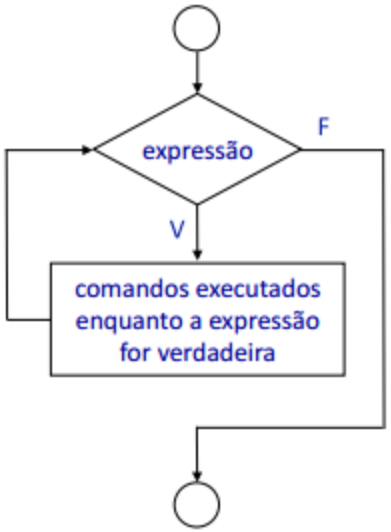
- Uma estrutura de repetição é utilizada quando um trecho de um algoritmo precisa ser executado diversas vezes.
- A cada execução do trecho, algumas variáveis terão seus valores modificados por algum cálculo ou alguma seqüência de instruções.
- O número de repetições pode ser fixo ou pode depender do valor de uma condição.
- Se o número de repetições for atingido ou se a condição de execução for falsa, o fluxo de execução é desviado para o primeiro comando após a estrutura de repetição.

Estruturas de repetição

- Pré teste

Tipos de estruturas de repetição

Teste no início



while()

Estrutura de repetição com teste no início.

Sintaxe:

```
while (expressão)
{
// comandos executados enquanto
// a expressão for verdadeira
}
```

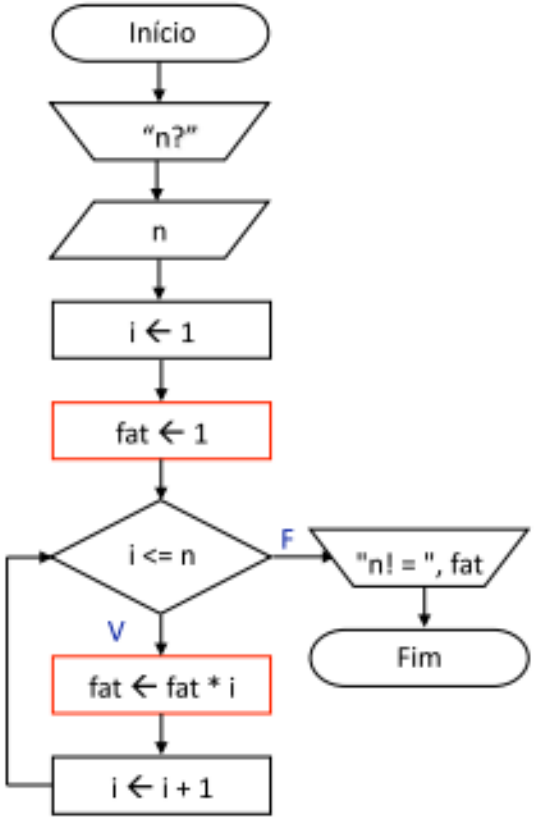
Atenção

Não se usa "**;**" após o **while()**.

Estruturas de repetição

- Pré teste

Escrever um programa para calcular n! = 1 × 2 × 3 × ... × n.



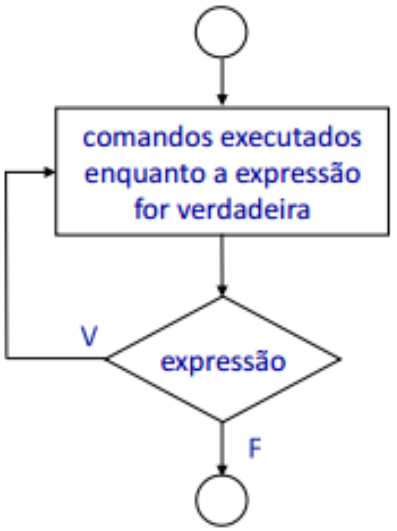

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int i, n;
  double fat;

  printf("Apresentar o fatorial de: ");
  scanf("%d", &n);
  i = 1;
  fat = 1;
  while ( i <= n )
  {
    fat = fat * i;
    i++;
  }
  printf("%d! = %.0f\n", n, fat);
  system("PAUSE");
  return(0);
}
```

Estruturas de repetição

- Pós teste

Tipos de estruturas de repetição

Teste no final



do-while()

Estrutura de repetição com teste no final.

Sintaxe:

```
do
{
// comandos executados enquanto
// a expressão for verdadeira
} while (expressão);
```

Atenção

Não se usa "**;**" após o **do**.

Estruturas de repetição

- Pós teste

Repetições com intervenção do usuário

```
int main()
{
  int num;
  char op;

  do
  {
    system("CLS");    // limpa a tela de execução

    printf("Informe um numero inteiro: ");
    scanf("%d", &num);

    if (num % 2 == 0)
      printf("%d é par\n", num);
    else
      printf("%d é ímpar\n", num);

    printf("Deseja executar novamente (S/N)? ");
    op = getchar();
  } while (op == 's' || op == 'S');

  system("PAUSE");
  return(0);
}
```

Repetição e seleção

Determinar o maior de um conjunto de 10 valores informados pelo usuário.

```
int main()
{
  float maior, valor;
  int cont;

  maior = -999999;
  cont = 0;
  while (cont < 10)
  {
    printf("Valor: ");
    scanf("%f", &valor);
    if (valor > maior)
      maior = valor;
    cont++;
  }
  printf("Maior valor informado: %f\n", maior);
  system("PAUSE");
  return(0);
}
```

Repetição dentro de repetição

Imprimir a tabuada do 1 ao 10.

```
int main()
{
  int a, b, valor;

  a = 1;
  while (a <= 10)
  {
    printf("\nTabuada do %d\n", a);
    b = 1;
    while (b <= 10)
    {
      valor = a * b;
      printf("%2d x %2d = %3d\n", a, b, valor);
      b++;
    }
    a++;
  }
  system("PAUSE");
  return(0);
}
```

Alguns Exercícios

O que será feito?

```
int main()
{
  float soma, termo;
  int cont, n;

  printf("Número de termos: ");
  scanf("%d", &n);
  cont = 0;
  soma = 0;
  termo = 0;

  do {
    cont++;
    soma = soma + termo;
    printf("Novo termo: ");
    scanf("%f", &termo);
  } while (condição);

  printf("Soma: %f (%d termos)\n", soma, cont);
  system("PAUSE");
  return(0);
}
```

Se condição for:

soma < 1000

Serão somados termos enquanto a soma não ultrapassar o valor 1000.

termo > 0

A soma considera apenas termos positivos.

cont < n

Serão somados exatamente n termos.

O que será impresso na linha 16

```
1  #include "stdio.h"
2  #include "stdlib.h"
3
4  main(){
5    int a = 1 , b = 2;
6
7    while(a < 16){
8        a += b;
9
10       do{
11         b += a;
12         a++;
13       }while(b < 9);
14   }
15
16   printf("a = %d, b = %d",a,b);
17   system("pause");
18 }
```

Controle do número de repetições

- A cada execução dos comandos de uma estrutura de repetição dá-se o nome de iteração.
- Em alguns problemas o número de iterações é fixo ou determinado em tempo de execução. Nestes casos, usa-se uma variável de controle do número de iterações.
- A variável de controle deve ser inicializada antes do primeiro comando da estrutura de repetição.
- Dentro da estrutura, a variável de controle terá seu valor atualizado de acordo com algum incremento ou decremento.
- A variável de controle será usada na expressão que determina se haverá a execução de uma nova iteração.

Controle no número de iterações

Exibir os *n* primeiros números ímpares positivos.

```
int main()
{
  int num, n, cont;

  printf("Quantos números ímpares deseja exibir? ");
  scanf("%d", &n);
  cont = 1;
  num = 1;

  while ( cont <= n )
  {
    printf("%d ", num);
    num = num + 2;
    cont++;
  }

  system("PAUSE");
  return(0);
}
```

variável de controle

inicialização

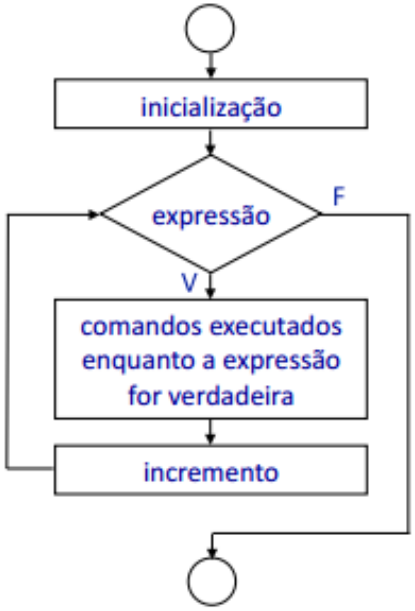
expressão

incremento

- Pré teste

Tipos de estruturas de repetição

Controle no número de repetições



for()

Estrutura de repetição com controle do número de repetições.

Sintaxe:

```
for(inicialização; expressão; incremento)
{
// comandos executados enquanto
// a expressão for verdadeira
}
```

Atenção

Não se usa "**;**" após o **for()**.

- Pré teste

Exemplo: estrutura de repetição `for( )`

Exibir os *n* primeiros números ímpares positivos.

```
int main()
{
  int num, n, cont;

  printf("Quantos números ímpares deseja exibir? ");
  scanf("%d", &n);
  num = 1;

  for ( cont = 1; cont <= n; cont++ )
  {
    printf("%d ", num);
    num = num + 2;
  }

  system("PAUSE");
  return(0);
}
```

variável de controle

inicialização

incremento

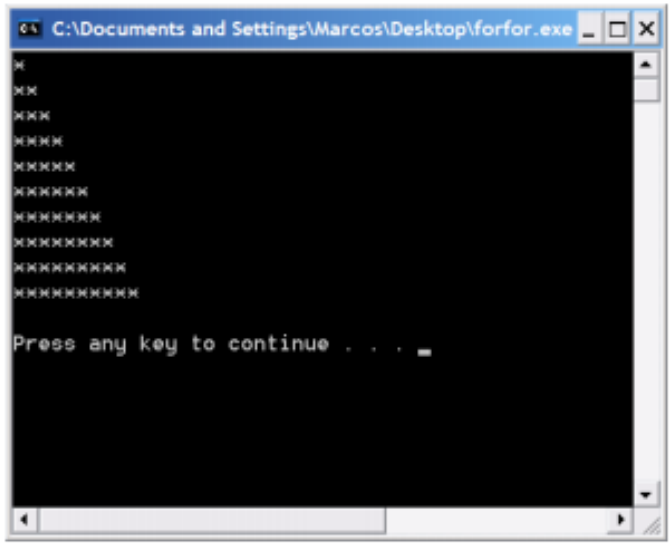
expressão

- Pré teste

Aninhamento: `for( )` dentro de `for( )` dentro de...

No aninhamento, todas as iterações do laço mais interno serão executadas antes da próxima iteração do laço mais externo.

```
int main()
{
  int i, j;

  for ( i = 1; i <= 10; i++ )
  {
    for ( j = 0; j < i; j++ )
    {
      printf("*");
    }
    printf("\n");
  }

  system("PAUSE");
  return(0);
}
```



Neste exemplo, a condição de execução do laço interno depende do valor da variável de controle do laço externo.