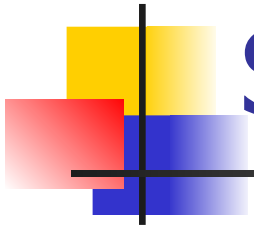
SCC0221 – Introdução à Ciência de Computação I

Prof.: Dr. Rudinei Goularte

(rudinei@icmc.usp.br)

Aulas 12 e 13 – Comandos de repetição em C

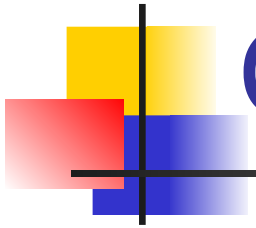
Instituto de Ciências Matemáticas e de Computação - ICMC
Sala 4-229



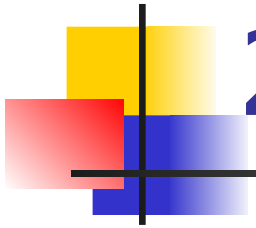
Sumário

- Iteração (Repetição).
  - for
  - while
  - do-while
- Desvio.
  - return
  - goto
  - break
  - continue

1. Tipos de Comandos para Controle de Fluxo

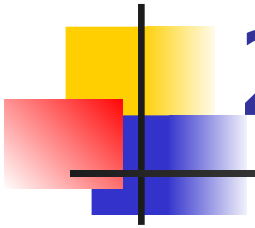


- Padrão ANSI divide os comandos C em:
  - Seleção
  - Iteração (repetição)
  - Desvio
  - Rótulo
  - Expressão
  - Bloco



2. Comandos de Iteração

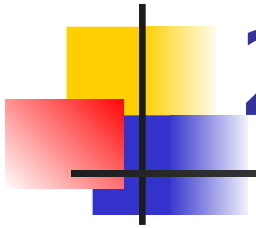
- Permitem que um conjunto de instruções seja executado, repetidas vezes, até que uma certa condição ocorra.
- O número de repetições pode ser pré-definido ou indeterminado.



2.1 O Comando for

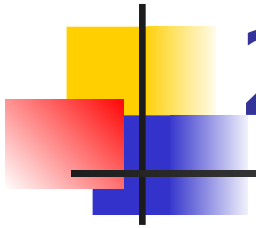
- Encontrado, de um modo ou de outro, em praticamente todas as linguagens.
- Em C, fornece maiores flexibilidade e capacidade.
- Forma geral:

for (*inicialização*; *condição*; *incremento***)**
comando;



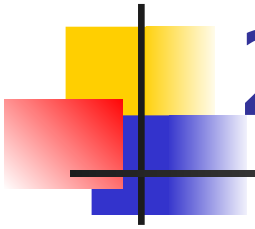
2.1 O Comando for

- Permite muitas variações!!!!
- De modo geral:
  - *inicialização*: é uma atribuição. Coloca um valor na variável de controle do laço.
  - *condição*: é uma expressão relacional. Determina o fim do laço – quando a condição for falsa.
  - *incremento*: determina como a variável de controle varia a cada iteração.
  - *comando*: pode ser vazio, simples ou um bloco.



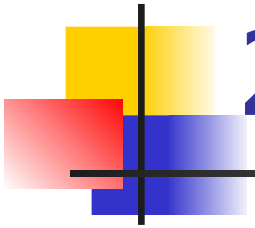
2.1 O Comando for

- As três seções – inicialização, condição e incremento - devem ser separadas por ponto-e-vírgula (;)
- Quando condição se torna falsa, programa continua execução na sentença seguinte ao for.



2.1 Exercícios

- Faça um comando for para contar de 1 até 100, de um em um.
- Faça um comando for para contar de 0 até 100, de 5 em 5.
- Faça um comando for para contar de 100 até 3, decrementando de um em um.

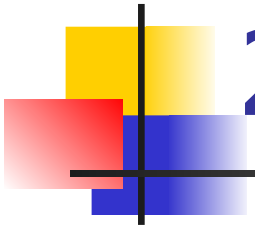


2.1 O Comando for

- Variação do for
  - Usar operador vírgula para controlar várias variáveis.

- O que faz o comando abaixo?

```
for (i = 0, j = 100; i + j >= 0; i ++, j -= 2)
    printf("%d ", i);
```

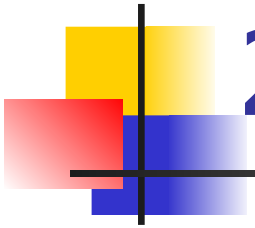


2.1 O Comando for

- As expressões (inicialização, condição, incremento e comando) são opcionais!

```
for (x = 0; x != 34; )
   scanf("%d", &x);
```

O quê acontece?



2.1 O Comando for

```
int x = 0;
for ( ; x != 34; )
   scanf("%d", &x);
```

- Aqui a inicialização foi realizada "fora" do for.

2.1 O Comando for

- Laço infinito.

```
for ( ; ; )
   printf("%c ", 'a');
```

- Laço sem corpo

```
for (x = 0 ; x < 1000; x++)
                ;
```

2.2 O Comando while

- Forma geral

  while(*condição***)**

  *comando***;**

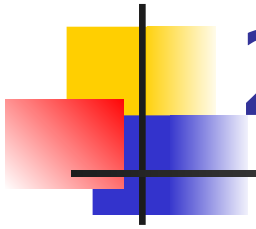
- *condição*: é qualquer expressão. Determina o fim do laço: quando a condição é falsa. Execução continua na sentença seguinte ao while.
- *comando*:pode ser vazio, simples ou um bloco.

2.2 O Comando while

- Assim como o for, o while testa uma *condição* antes de entrar no laço.

```
char ch ='\0'; /*caracter nulo*/

while (ch != 'Z')
   scanf("%c", &ch);
printf ("Z: fim do laço while");
```



2.2 Exercício

- Escreva um programa, usando while, para calcular o fatorial de um número n tomado como entrada do usuário.

2.3 O Comando do-while

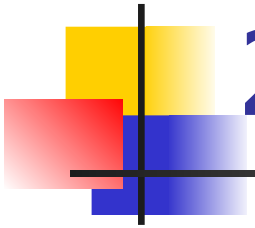
- Forma geral

do{

comando ;

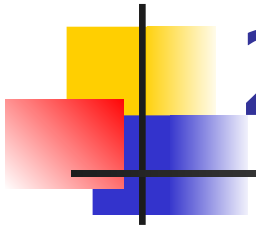
}**while(***condição***);**

- *comando*: pode ser vazio, simples ou um bloco.
- *condição*: pode ser qualquer expressão. Se falsa, o comando é terminado e a execução continua na sentença seguinte ao do-while.



2.3 O Comando do-while

- Diferente de *for* e de *while*, o comando *do-while* testa a *condição* no fim do laço.
  - O *comando* será executado, pelo menos, uma vez.
  - As chaves não são necessárias em *comandos* simples. Porém, evitam confusão com o *while*.
    - Boa prática de programação!



2.3 Exercício

- Uso mais comum do *do-while* é em rotinas de seleção de um menu. Desenvolva uma calculadora básica, usando *do-while* e *switch*, que possua o seguinte menu:

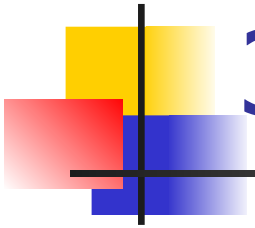
  +: adição

  -: subtração

  *: multiplicação

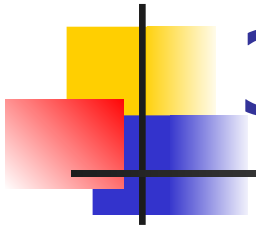
  /: divisão

Exemplo de saída: `O resultado de 3.00 * 5.00 é 15.00`



3. Comandos de Desvio

- C possui 4 comandos para desvio incondicional: *return*, *goto*, *break* e *continue*.
- *return* e *goto* podem ser usados em qualquer lugar no programa.
- *break* e *continue* podem ser usados em conjunto com qualquer comando de iteração.
- *break* pode ser usado em conjunto com *switch*.

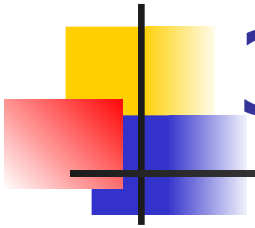


3.1 O Comando return

- Forma geral

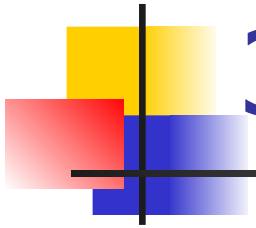
  return *expressão*;

- Usado para retornar de funções. Se *return* tem um valor associado, esse é o valor de retorno da função.
  - Caso contrário, lixo ou zero (em alguns compiladores).
- A expressão é opcional.



3.2 O Comando goto

- Tendência a tornar programas ilegíveis.
- C possui rico conjunto de comandos de controle. Aliados a break e continue, implicam em pouca necessidade de uso para goto.
- Existem casos onde seu uso é passível :: prudência máxima!!!!
- **Nessa disciplina não usaremos goto!!!**



3.2 O Comando goto

- Forma geral:

  goto *rótulo*;

  .

  .

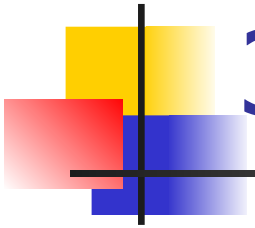
  .

  rótulo:

- *rótulo* pode estar em qualquer lugar no programa.

3.2 O Comando goto

```
x = 1;
loop:
   x++;
   if (x < 100)
         goto loop;
printf("%d. Poderia ter usado um for!", x);
```



3.3 O Comando break

- Dois usos:
  - Terminar um case em um comando switch.

```
int x;
scanf("%d", &x);
switch(x){
    case 1:
    case 2:
    case 3: printf ("3"); break;
    case 4: printf ("4");
    case 5: printf ("5"); break;
}
```

3.3 O Comando break

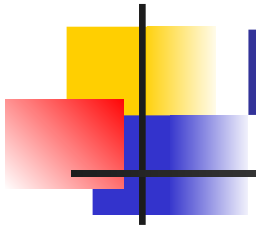
- Forçar a terminação imediata de um laço (o mais interno).

```
int x;
for (x = 1; x < 100; x++)
{
        printf("%d ", x);
        if (x == 13)
                break;
}
```

3.4 O Comando continue

- Força que ocorra a próxima iteração do laço.

```
for (i = 0; i < 100; i++){
    if ((i % 2)== 0)
            continue;
        printf("%d ", i);
}
```

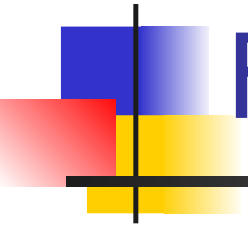


Exercício

Desenvolva um programa em C que calcule a soma dos 20 primeiros termos da série:

$S = X - (X^2/3!) + (X^4/5!) - (X^6/7!) + (X^8/9!) - \ldots$

O valor de X deve ser fornecido pelo usuário.



Fim.