

Controle do Programa

PE-04 – v1.0

Prof. Paulo Joia Filho

- 1 Introdução
- 2 Estruturas de controle
- 3 Comandos de controle
- 4 Conclusões

- 1 **Introdução**
 - Objetivos
- 2 Estruturas de controle
- 3 Comandos de controle
- 4 Conclusões

Objetivos

Os principais objetivos desta aula são:

- Explorar as estruturas de controle da linguagem C.
- Praticar através dos exemplos e exercícios em sala.

1 Introdução

2 Estruturas de controle

- Programação estruturada
- Estruturas de sequência
- Estruturas de decisão
- Estruturas de repetição

3 Comandos de controle

4 Conclusões

Programação Estruturada (PE)

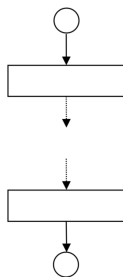
O paradigma da programação estruturada (ou procedural), afirma que qualquer problema pode ser quebrado em problemas menores, de mais fácil solução, chamados de funções, sub-rotinas ou procedimentos.

Além disso...

- O paradigma estruturado preconiza que todos os processamentos possíveis podem ser reduzidos a apenas três tipos de estruturas:
 - ❖ Estruturas de sequência;
 - ❖ Estruturas de decisão (ou condicionais);
 - ❖ Estruturas de repetição (ou iterativas).

Estruturas de sequência

- O fluxo de execução das instruções ocorre de modo linear, uma após a outra.
- Deve existir apenas um caminho possível no conjunto de instruções de um algoritmo.



Seqüência

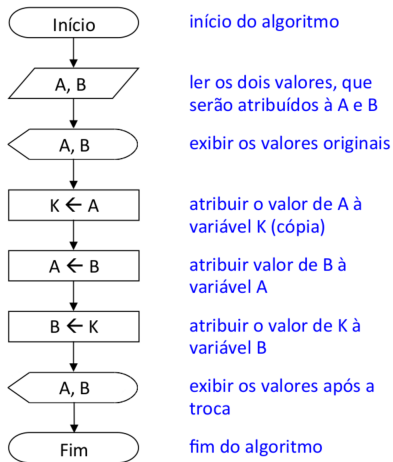
Exemplo de sequência

Ler e exibir dois valores numéricos A e B, depois trocar os valores entre si e exibir novamente.

Sejam:

A: o primeiro valor numérico.

B: o segundo valor numérico.



Exemplo de sequência

Implementação em C

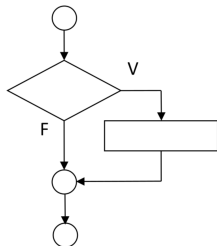
[exemplo_sequencia.c]

```
1 #include <stdio.h>
3 int main() {
    // Início do programa
5     printf("Exemplo de Sequência\n\n");
    int A, B, K;
7
    // Ler os valores que serão atribuídos a A e B
9     printf("Informe um valor inteiro para A: ");
    scanf("%i", &A);
11    printf("Informe um valor inteiro para B: ");
    scanf("%i", &B);
13
    // Exibir os valores originais de A e B
15    printf("Valores originais: \n");
    printf("A = %i\n", A);
17    printf("B = %i\n", B);
```

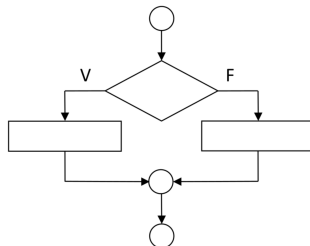
```
    // Atribuir o valor de A à variável K (cópia)
20    K = A;
22    // Atribuir o valor de B à variável A
    A = B;
24
    // Atribuir o valor de K à variável B
26    B = K;
28    // Exibir os valores de A e B após a troca
    printf("Valores após a troca: \n");
30    printf("A = %i\n", A);
    printf("B = %i\n", B);
32
    // Fim do programa
34    return 0;
}
```

Estruturas de decisão

- Execução seletiva de um grupo de instruções baseada em alguma condição.
 - ▶ Normalmente uma expressão lógica ou relacional.
- Criam alternativas no fluxo de execução do algoritmo.
 - ▶ Durante a execução, apenas uma das alternativas será escolhida.



Seleção simples



Seleção composta

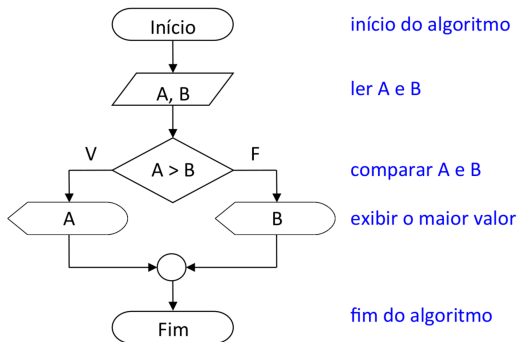
Exemplo de decisão

Ler dois valores numéricos, compará-los e exibir o maior valor.

Sejam:

A: o primeiro valor numérico.

B: o segundo valor numérico.



Exemplo de decisão

Implementação em C

[exemplo_decisao.c]

```
1  #include <stdio.h>

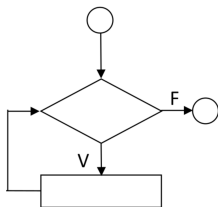
3  int main() {
    printf("Exemplo de Decisão\n\n");
5     float A, B;

7     // Ler os valores que serão atribuídos a A e B
    printf("Informe um valor para A: ");
9     scanf("%f", &A);
    printf("Informe um valor para B: ");
11    scanf("%f", &B);

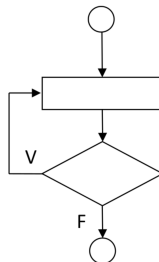
13    // Comparar A e B
    if (A > B) {
15        printf("O maior valor é A = %g\n", A);
    } else {
17        printf("O maior valor (ou igual) é B = %g\n", B);
    }
19    return 0;
}
```

Estruturas de repetição

- Execução seletiva de um grupo de instruções até que alguma condição seja satisfeita.
- O fluxo da execução pode realizar várias repetições de um mesmo conjunto de comandos, antes de prosseguir para a etapa seguinte.



**Iteração com
teste *a priori***



**Iteração com
teste *a posteriori***

Exemplo de repetição

Calcular o resto da divisão inteira entre dois números inteiros positivos.

Sejam:

A: o valor do dividendo.

B: o valor do divisor.

Q: o valor do quociente.

R: o valor do resto.

início do algoritmo

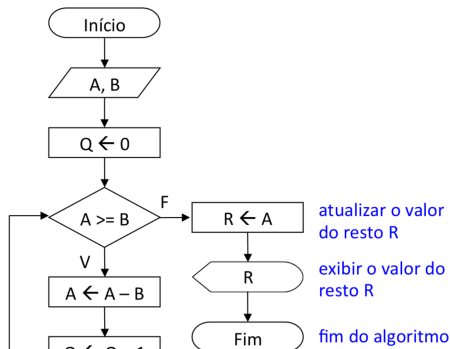
ler A e B

inicializar o valor
do quociente Q

comparar A e B

atualizar o valor do
dividendo A

atualizar o valor do
quociente Q e retornar



Exemplo de repetição

Implementação em C

[exemplo_repeticao.c]

```
1 #include <stdio.h>

3 int main() {
    printf("Exemplo de Repetição\n\n");
5     int A, B, Q, R;

7     // Ler os valores que serão atribuídos a A e B
    printf("Informe um valor inteiro para A: ");
9     scanf("%i", &A);
    printf("Informe um valor inteiro para B: ");
11    scanf("%i", &B);

13    // Inicializar o valor do quociente Q
    Q = 0;
```

```
16    // Comparar A e B
    while (A >= B) {
18        // Atualizar o valor do dividendo A
        A -= B;
20        // Atualizar o valor do quociente Q
        Q += 1;
22    }

24    // Atualizar o valor do resto R
    R = A;
26

28    // Exibir o valor do resto R
    printf("O valor do resto da divisão é R
        = %d\n", R);

30    return 0;
}
```

- 1 Introdução
- 2 Estruturas de controle
- 3 Comandos de controle**
 - Comandos de decisão
 - Comandos de repetição
- 4 Conclusões

Comandos de decisão

if, switch e operador ternário

- Comandos de decisão (ou condicionais) são aqueles que dependendo de uma condição executam um bloco, caso a condição não seja atendida, o bloco não será executado.
- C provê suporte a três comandos de decisão:
 - `if`
 - `switch`
 - operador ternário (`? :`)

Comandos de decisão

Sintaxe do comando `if`

```
if (condicao) {  
    comando1;  
    comando2;  
    comando3;  
}
```

Observações:

- Os parênteses que envolvem a condição são **obrigatórios**.
- A condição deverá retornar um valor lógico (booleano).
- Os comandos somente serão executados se a condição for **verdadeira**.

Comandos de decisão

Comando `if`

- O uso de chaves **não** é obrigatório caso seja apenas um único comando.
- Porém, a boa prática recomenda a utilização de chaves independente do número de comandos.
 - Melhor indentação do código

```
if ( 1 ) comando;
```

equivale a:

```
if ( 1 ) {  
    comando;  
}
```

Comandos de decisão

A estrutura `if...else`

Sintaxe:

```
if (condicao) {  
    comando1;  
    comando2;  
    comando3;  
} else {  
    comando4;  
    comando5;  
    comando6;  
}
```

Comandos de decisão

[conceito.c]

A estrutura `if...else if...else`

```
1  #include<stdio.h>

3  int main() {
    float nota;

5     printf(" Informe a nota: ");
7     scanf("%f", &nota);

9     if (nota >= 9) {
        printf(" Conceito A \n");
11    } else if (nota >= 8) {
        printf(" Conceito B \n");
13    } else if (nota >= 7) {
        printf(" Conceito C \n");
15    } else {
        printf(" Reprovado \n");
17    }
    return 0;
19 }
```

- Qual a saída quando $nota = 6.999999$?
- E se $nota = 6.9999999$?

Comandos de decisão

[conceito.c]

A estrutura `if...else if...else`

```
1  #include <stdio.h>

3  int main() {
    float nota;

5     printf(" Informe a nota: ");
7     scanf("%f", &nota);

9     if (nota >= 9) {
        printf(" Conceito A \n");
11    } else if (nota >= 8) {
        printf(" Conceito B \n");
13    } else if (nota >= 7) {
        printf(" Conceito C \n");
15    } else {
        printf(" Reprovado \n");
17    }

19    return 0;
}
```

- Qual a saída quando $nota = 6.999999$?
- E se $nota = 6.9999999$?

```
Informe a nota: 6.999999
Reprovado
```

```
Informe a nota: 6.9999999
Conceito C
```

Por quê?

Comandos de decisão

Comando `switch`

- Na instrução `switch`, uma variável de tipo primitivo `char` ou `int` é comparada com cada valor em questão. Se um valor coincidente é achado, a instrução (ou instruções) **depois do teste** é executada.
- Sintaxe:

```
switch (variavel) {  
    case 1: comandoA; break;  
    case 2: comandoB; break;  
    case 3: comandoC; break;  
    default: comandoPadrao;  
}
```

Comandos de decisão

Comando `switch`

- Se nenhum valor for encontrado, a instrução **default** é executada.
- O comando **break** é necessário para quebrar o **switch**, pois assim que encontrada a opção correta, é executado tudo em seguida.

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5

    switch (letra) {
7        case 'A': printf(" Entrou em A \n"); break;
        case 'B': printf(" Entrou em B \n");
9        case 'C': printf(" Entrou em C \n"); break;
        case 'D': printf(" Entrou em D \n"); break;
11       default: printf(" Entrou em Default \n");
    }
13   return 0;
}
```

- E se o valor de letra for 'C'?

- E se for 'a'?

- Qual a saída deste programa?

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5

    switch (letra) {
7      case 'A': printf(" Entrou em A \n"); break;
      case 'B': printf(" Entrou em B \n");
9      case 'C': printf(" Entrou em C \n"); break;
      case 'D': printf(" Entrou em D \n"); break;
11     default: printf(" Entrou em Default \n");
    }
13     return 0;
}
```

- E se o valor de letra for 'C'?

- E se for 'a'?

- Qual a saída deste programa?

```
Entrou em B
Entrou em C
```

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5

    switch (letra) {
7        case 'A': printf(" Entrou em A \n"); break;
        case 'B': printf(" Entrou em B \n");
9        case 'C': printf(" Entrou em C \n"); break;
        case 'D': printf(" Entrou em D \n"); break;
11       default: printf(" Entrou em Default \n");
    }
13   return 0;
}
```

- E se o valor de letra for 'C'?

- E se for 'a'?

- Qual a saída deste programa?

```
Entrou em B
Entrou em C
```

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5     switch (letra) {
7         case 'A': printf(" Entrou em A \n"); break;
        case 'B': printf(" Entrou em B \n");
9         case 'C': printf(" Entrou em C \n"); break;
        case 'D': printf(" Entrou em D \n"); break;
11        default: printf(" Entrou em Default \n");
    }
13    return 0;
}
```

- E se o valor de letra for 'C'?

Entrou em C

- E se for 'a'?

- Qual a saída deste programa?

Entrou em B
Entrou em C

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5     switch (letra) {
7         case 'A': printf(" Entrou em A \n"); break;
        case 'B': printf(" Entrou em B \n");
9         case 'C': printf(" Entrou em C \n"); break;
        case 'D': printf(" Entrou em D \n"); break;
11        default: printf(" Entrou em Default \n");
    }
13    return 0;
}
```

- E se o valor de letra for 'C'?

Entrou em C

- E se for 'a'?

- Qual a saída deste programa?

Entrou em B
Entrou em C

Comandos de decisão

Comando switch

[switch1.c]

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5     switch (letra) {
7         case 'A': printf(" Entrou em A \n"); break;
        case 'B': printf(" Entrou em B \n");
9         case 'C': printf(" Entrou em C \n"); break;
        case 'D': printf(" Entrou em D \n"); break;
11        default: printf(" Entrou em Default \n");
    }
13    return 0;
}
```

- E se o valor de letra for 'C'?

Entrou em C

- E se for 'a'?

Entrou em Default

- Qual a saída deste programa?

Entrou em B
Entrou em C

Comandos de decisão

[switch2.c]

Comando switch

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5     switch (letra) {
6         default: printf(" Entrou em Default \n");
7         case 'A': printf(" Entrou em A \n"); break;
8         case 'B': printf(" Entrou em B \n"); break;
9         case 'C': printf(" Entrou em C \n"); break;
10        case 'D': printf(" Entrou em D \n"); break;
11    }
12    return 0;
13 }
```

- Qual a saída deste programa?

Comandos de decisão

[switch2.c]

Comando switch

```
1  #include<stdio.h>

3  int main() {
    char letra = 'B';

5      switch (letra) {
6          default: printf(" Entrou em Default \n");
7          case 'A': printf(" Entrou em A \n"); break;
8          case 'B': printf(" Entrou em B \n"); break;
9          case 'C': printf(" Entrou em C \n"); break;
10         case 'D': printf(" Entrou em D \n"); break;
11     }
12     return 0;
13 }
```

- Qual a saída deste programa?

Entrou em B

Comandos de decisão

[switch3.c]

Comando switch

```
1  #include<stdio.h>

3  int main() {
    char letra = 'b';

5     switch (letra) {
6         case 'A': printf(" Entrou em A \n"); break;
7         case 'B': printf(" Entrou em B \n"); break;
8         default: printf(" Entrou em Default \n");
9         case 'C': printf(" Entrou em C \n"); break;
10        case 'D': printf(" Entrou em D \n"); break;
11    }
12    return 0;
13 }
```

- Qual a saída deste programa?

Comandos de decisão

[switch3.c]

Comando switch

```
1  #include<stdio.h>

3  int main() {
    char letra = 'b';

5     switch (letra) {
6         case 'A': printf(" Entrou em A \n"); break;
7         case 'B': printf(" Entrou em B \n"); break;
8         default: printf(" Entrou em Default \n");
9         case 'C': printf(" Entrou em C \n"); break;
10        case 'D': printf(" Entrou em D \n"); break;
11    }
12    return 0;
13 }
```

- Qual a saída deste programa?

```
Entrou em Default
Entrou em C
```

Comandos de decisão

Operador ternário

- Existem situações cujo uso do `if` não é “elegante”.
- Por exemplo, suponha que a função `max` retorne o maior número dentre os dois passados via parâmetros formais:

```
1      int max (int a, int b) {  
        if (a > b) {  
3          return a;  
        } else {  
5          return b;  
        }  
7      }
```

Comandos de decisão

Operador ternário

- O operador ternário é uma expressão, significando que ele devolve um valor.
- O operador ternário é muito útil para condicionais (curtas e simples) e tem o seguinte formato:

```
variável = <condição> ? seTrue : seFalse;
```

- A condição pode estar envolvida entre parênteses para facilitar a leitura, contudo não é obrigatório

Comandos de decisão

Operador ternário

- Exemplos:

```
int a = 2;  
int b = 3;  
int c = (a > b) ? a : b;
```

- Indica que se **a** for maior que **b**, **c** recebe o valor de **a**, caso contrário recebe o valor de **a**, isto é, **c** recebe o maior valor entre **a** e **b**.
- Qual o valor das variáveis abaixo?

```
int peso = (2 != 2) ? 80 : 63;
```

```
char letra = (1 == 1) ? 'R' : 'T';
```

Comandos de decisão

Operador ternário

- Não necessariamente o retorno do operador ternário deve ser atribuído a uma variável.
- Por exemplo, seu retorno pode ser o retorno de uma função como faz a função **max** com operador ternário:

```
int max(int a, int b) {  
    return (a > b) ? a : b;  
}
```

- Ou mesmo o retorno pode servir como parâmetro de chamada de uma função:

```
printf ((a > b) ? "a maior!" : "b maior!");  
printf ("%d", (a > b) ? a : b);
```

Comandos de repetição

Comandos `while`, `do...while` e `for`

- Comandos de repetição são utilizados para repetir um bloco de código.
- C provê suporte a três comandos de repetição:
 - `while` [enquanto]
 - `do...while` [faça...enquanto]
 - `for` [para]

Comandos de repetição

Comando `while`

- O comando `while` é utilizado para repetir um bloco de acordo com uma condição.
- É considerado um *loop* de pré-teste.
 - Isto é, testa a condição antes de executar o bloco.
- Sintaxe:

```
while (condicao) {  
    comando1;  
    comando2;  
    comandoN;  
}
```


Comandos de repetição

[while1.c]

Exemplo com while

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5     while (i < 10) {
7         printf(" %d", ++i);
        }
9     printf("\n");
    return 0;
11 }
```

- Qual a saída deste programa?

Comandos de repetição

[while1.c]

Exemplo com while

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5     while (i < 10) {
7         printf(" %d", ++i);
        }
9     printf("\n");
    return 0;
11 }
```

- Qual a saída deste programa?

```
1 2 3 4 5 6 7 8 9 10
```

Comandos de repetição

Comando `do...while`

- O comando `do...while` é semelhante ao `while`, contudo é um comando de repetição de pós-teste.
 - Isto é, somente ao final da execução do bloco que se verifica a condição.
- Geralmente, é utilizado quando se deseja testar a condição somente a partir da segunda iteração.
 - Por exemplo, a leitura da opção de um menu. Neste caso, a primeira vez é solicitada a digitação de uma opção. **Somente** se a opção digitada for inválida é solicitado para digitar novamente.

Comandos de decisão

Sintaxe do comando `do...while`

```
do {  
    comando1;  
    comando2;  
    comandoN;  
} while (condicao);
```

👉 **Observação:** não esquecer o ponto-e-vírgula após a condição.

Comandos de repetição

Exemplo com do...while

[while2.c]

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5
    do {
7      printf("\t%d", ++i);
    } while (i != 1);

9
    printf("\n");
11   return 0;
}
```

- Qual a saída?

Comandos de repetição

[while2.c]

Exemplo com do...while

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5     do {
7         printf("\t%d", ++i);
        } while (i != 1);

9     printf("\n");
11    return 0;
}
```

- Qual a saída?

A black rectangular box representing a terminal window, containing the number '1' in a light blue monospaced font.

Comandos de repetição

[while3.c]

Outro exemplo com `do...while`

```
1  #include <stdio.h>

3  int main() {
    int i;

5     do {
7         printf(" Digite um inteiro entre 0 e 10: ");
        scanf("%d", &i);
9     } while (i < 0 || i > 10);

11    printf(" Numero digitado: %d\n", i);
    return 0;
13 }
```

- Qual a saída quando $i = -1$, $i = 11$ e $i = 5$?

Comandos de repetição

[while3.c]

Outro exemplo com do...while

```
1  #include <stdio.h>

3  int main() {
    int i;

5     do {
7         printf(" Digite um inteiro entre 0 e 10: ");
        scanf("%d", &i);
9     } while (i < 0 || i > 10);

11    printf(" Numero digitado: %d\n", i);
    return 0;

13 }
```

- Qual a saída quando $i = -1$, $i = 11$ e $i = 5$?

```
Digite um inteiro entre 0 e 10: -1
Digite um inteiro entre 0 e 10: 11
Digite um inteiro entre 0 e 10: 5
Numero digitado: 5
```


Comandos de repetição

Comando `for`

- Comando de repetição mais versátil da linguagem C.
- É composto por:
 - **Inicialização**: executado uma única vez no início do *loop*.
 - **Condição**: executado sempre antes de cada iteração. Se verdadeira, o bloco é executado. Se falsa, é finalizado.
 - **Operação**: executado sempre ao término de cada iteração.
- Sintaxe:

```
for (inicializacao; condicao; operacao) {  
    comando1;  
    comando2;  
    ...  
    comandoN;  
}
```

Comandos de repetição

[for1.c, for2.c]

Exemplo com for

```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
7         printf(" %d", i);
        }

9     printf("\n");
11    return 0;
}
```

- Qual a saída?

Comandos de repetição

[for1.c, for2.c]

Exemplo com for

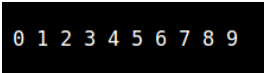
```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
7         printf(" %d", i);
        }

9     printf("\n");
11    return 0;
}
```

- Qual a saída?



0 1 2 3 4 5 6 7 8 9

Comandos de repetição

Exemplo com for

```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
7         printf(" %d", i);
        }

9     printf("\n");
11    return 0;
}
```

- Qual a saída?

```
0 1 2 3 4 5 6 7 8 9
```

[for1.c, for2.c]

```
1  #include <stdio.h>

3  int main() {
    /* declarando a variável i
       dentro do próprio for */
5     for (int i = 0; i < 10; i++) {
7         printf(" %d", i);
        }

9     printf("\n");
11    return 0;
}
```

Comandos de repetição

Exemplo com for

```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
6         printf("%d", i);
7     }

9     printf("\n");
11    return 0;
}
```

- Qual a saída?

```
0 1 2 3 4 5 6 7 8 9
```

[for1.c, for2.c]

```
1  #include <stdio.h>

3  int main() {
    /* declarando a variável i
     dentro do próprio for */
5     for (int i = 0; i < 10; i++) {
6         printf(" %d", i);
7     }

9     printf("\n");
11    return 0;
}
```

- Tente compilar com o padrão C89.
O que acontece?

Comandos de repetição

Exemplo com for

```

1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
6         printf("%d", i);
7     }

9     printf("\n");
11    return 0;
}
```

- Qual a saída?

```
0 1 2 3 4 5 6 7 8 9
```

[for1.c, for2.c]

```

1  #include <stdio.h>

3  int main() {
    /* declarando a variável i
       dentro do próprio for */
5     for (int i = 0; i < 10; i++) {
6         printf(" %d", i);
7     }

9     printf("\n");
11    return 0;
}
```

- Tente compilar com o padrão C89. O que acontece?

```

for2.c: In function 'main':
for2.c:6:2: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
for (int i = 0; i < 10; i++) {
^
```

Comandos de repetição

[for3.c]

Comando `for`

No laço **`for`**, a **inicialização**, **condição** e **operação** são todas opcionais.

```
1  #include <stdio.h>

3  int main() {
    unsigned short int i = 0, j = 6e3;

5     for (;;) { /* Sem condicao, admite-se sempre verdade */
7         if (i % j == 0) {
            printf("\t%d\n", i);
9         }
            i++;
11     }
    return 0;
13 }
```

- Qual a saída deste programa?
- Por que isso acontece?

Comandos de repetição

[for3.c]

Comando for

No laço **for**, a **inicialização**, **condição** e **operação** são todas opcionais.

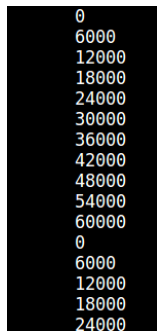
```
1  #include <stdio.h>

3  int main() {
    unsigned short int i = 0, j = 6e3;

5     for (;;) { /* Sem condicao, admite-se sempre verdade */
7         if (i % j == 0) {
            printf("\t%d\n", i);
9         }
            i++;
11     }
    return 0;
13 }
```

● Qual a saída deste programa?

● Por que isso acontece?



Executa infinitamente...

Comandos de repetição

[for3.c]

Comando for

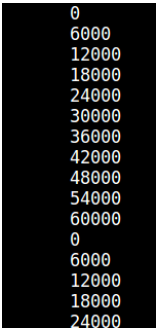
No laço **for**, a **inicialização**, **condição** e **operação** são todas opcionais.

```
1  #include <stdio.h>

3  int main() {
    unsigned short int i = 0, j = 6e3;

5     for (;;) { /* Sem condicao, admite-se sempre verdade */
7         if (i % j == 0) {
            printf("\t%d\n", i);
9         }
        i++;
11     }
    return 0;
13 }
```

- Qual a saída deste programa?
- Por que isso acontece?



0
6000
12000
18000
24000
30000
36000
42000
48000
54000
60000
0
6000
12000
18000
24000

Executa infinitamente...

Overflow em inteiros sem sinal

Erro de *wraparound*

wraparound

[**processamento de dados**] s. recorrente (operação), f.; continuação de operação do último passa para o primeiro a fecha o ciclo

C's unsigned integer types are “modulo” in the sense in that overflows or out-of-bounds results **silently wrap**.

[ISO/IEC 9899:TC3, 2007]

Integer overflow is the canonical example of “undefined behaviour” in C (notice that operations on **unsigned integers** never overflow, they are defined to **wraparound** instead). This means that once you've executed $x + y$, if it overflowed, you're already hosed. It's too late to do any checking: your program could have crashed already.

[https://stackoverflow.com/questions/2633661/
how-to-check-integer-overflow-in-c](https://stackoverflow.com/questions/2633661/how-to-check-integer-overflow-in-c)

Comandos de repetição

[for4.c]

O Comando `for` pode ser aninhado

Podemos ter um `for` dentro de outro, e dentro de outro, e dentro de outro...

```
1  #include <stdio.h>

3  int main() {
    int i, j;

5     for (i = 0; i <= 2; i++) {
7         for (j = 0; j < 2; j++) {
            printf(" i = %d;\t j = %d\n", i, j);
9         }
    }

11  return 0;
}
```

- Qual a saída?

Comandos de repetição

[for4.c]

O Comando `for` pode ser aninhado

Podemos ter um `for` dentro de outro, e dentro de outro, e dentro de outro...

```
1  #include <stdio.h>

3  int main() {
    int i, j;

5     for (i = 0; i <= 2; i++) {
7         for (j = 0; j < 2; j++) {
            printf(" i = %d;\t j = %d\n", i, j);
9         }
    }

11  return 0;
}
```

- Qual a saída?

```
i = 0;   j = 0
i = 0;   j = 1
i = 1;   j = 0
i = 1;   j = 1
i = 2;   j = 0
i = 2;   j = 1
```

Comandos de repetição

[for5.c]

Comando `break`

- Inserido dentro de um bloco de repetição;
 - ☞ também funciona para `while` e `do...while`.
- Caso seja executado, o bloco de repetição é finalizado.

Qual a saída do programa abaixo?

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5      for (; i < 10; i++) {
6          if (i == 3) {
7              break;
8          }
9          printf(" %d\n", i);
10     }
11     return 0;
12 }
13 }
```

Comandos de repetição

[for5.c]

Comando `break`

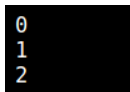
- Inserido dentro de um bloco de repetição;
 - ☞ também funciona para `while` e `do...while`.
- Caso seja executado, o bloco de repetição é finalizado.

Qual a saída do programa abaixo?

```
1  #include <stdio.h>

3  int main() {
    int i = 0;

5      for (; i < 10; i++) {
6          if (i == 3) {
7              break;
8          }
9          printf(" %d\n", i);
11     }
12     return 0;
13 }
```



```
0
1
2
```

Comandos de repetição

[for6.c]

Comando `continue`

- Inserido dentro de um bloco de repetição.
- Caso seja executado, a iteração atual do bloco de repetição é interrompida, seguindo para a próxima iteração.

Qual a saída do programa abaixo?

```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
7         if (i > 7 || i == 3) {
            continue;
9         }
        printf(" %d\n", i);
11     }
    return 0;
13 }
```

Comandos de repetição

[for6.c]

Comando `continue`

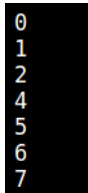
- Inserido dentro de um bloco de repetição.
- Caso seja executado, a iteração atual do bloco de repetição é interrompida, seguindo para a próxima iteração.

Qual a saída do programa abaixo?

```
1  #include <stdio.h>

3  int main() {
    int i;

5     for (i = 0; i < 10; i++) {
7         if (i > 7 || i == 3) {
            continue;
9         }
        printf(" %d\n", i);
11     }
    return 0;
13 }
```



0
1
2
4
5
6
7

Note a ausência do 3 e dos valores acima de 7.

- 1 Introdução
- 2 Estruturas de controle
- 3 Comandos de controle
- 4 Conclusões**
 - Exercícios de aprendizagem
 - Referências bibliográficas

Exercícios de Aprendizagem

Exercício 1

Elaborar um programa para ler quatro notas, calcular a média e informar se o aluno passou na disciplina (aprovado) ou não (reprovado). A média para passar deve ser igual ou superior a 6.

Exercício 2

Elaborar um programa para calcular o valor da função

$$f(x) = \begin{cases} x^2 + 16, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases}$$

para um valor de x fornecido pelo usuário.

Exercícios de Aprendizagem

Exercício 3

Escreva um programa para calcular o valor da função $f(x, y)$ para quaisquer valores reais de x e y fornecidos pelo usuário, tal que:

$$f(x, y) = \begin{cases} x + y, & \text{se } x \geq 0 \text{ e } y \geq 0 \\ x + y^2, & \text{se } x \geq 0 \text{ e } y < 0 \\ x^2 + y, & \text{se } x < 0 \text{ e } y \geq 0 \\ x^2 + y^2, & \text{se } x < 0 \text{ e } y < 0 \end{cases}$$

Exercício 4

Elabore um programa que calcule e escreva o valor da soma abaixo:

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \cdots + \frac{99}{50}$$

Exercícios de Aprendizagem

O termo geral de uma Progressão Geométrica (PG) de razão q é dado por:

$$a_n = a_1 q^{n-1},$$

onde a_1 é o primeiro termo da série.

Exercício 5

Dada uma PG de razão 4 e primeiro termo igual a 3, faça um programa, usando **um único laço de repetição**, para retornar:

- Os 12 primeiros termos desta série;
- A soma dos 5 primeiros termos.

Referências Bibliográficas I



Aguilar, L. J. (2008).

Programação em C++: Algoritmos, Estruturas de Dados e Objetos.
McGraw-Hill, São Paulo.



Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2002).

Algoritmos: Teoria e Prática.
Elsevier, Rio de Janeiro.



Drozdek, A. (2009).

Estrutura de Dados e Algoritmos em C++.
Cengage Learning, São Paulo.



Forbellone, A. L. V. e Eberspacher, H. F. (2005).

Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados.
Pearson Prentice Hall, São Paulo, 3 edition.



ISO/IEC 9899:TC3 (2007).

Programming Languages - C: International Standard.

International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).

Disponível em: <<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf>>. Acesso em: 2018-09-16.

Referências Bibliográficas II



Knuth, D. E. (2005).

The Art of Computer Programming.

Addison-Wesley, Upper Saddle River, NJ, USA.



Pinheiro, F. d. A. C. (2012).

Elementos de Programação em C.

Bookman, Porto Alegre.



Sedgewick, R. (1998).

Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, Searching.

Addison-Wesley, Boston, 3rd edition.



Szwarcfiter, J. L. e Markenzon, L. (1994).

Estruturas de Dados e Seus Algoritmos.

LTC, Rio de Janeiro.



Tenenbaum, A. A., Langsam, Y., e Augenstein, M. J. (1995).

Estruturas de Dados Usando C.

Makron Books, São Paulo.

Referências Bibliográficas III



Terra, R. (2014).

Linguagem C - Notas de Aula.

Universidade Federal de Lavras (UFLA).

Disponível em:

http://professores.dcc.ufla.br/~terra/public_files/2014_apostila_c_ansi.pdf. Acesso em: 2018-09-16.