

Estruturas de Dados

Módulo 3 – Controle de Fluxo

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO



Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel,
Introdução a Estruturas de Dados, Editora Campus
(2004)

Capítulo 3 – Controle de fluxo

Tópicos

- Tomada de decisão
- Construções com laços
- Seleção

Tomada de Decisão

- Exemplo:
 - função para qualificar a temperatura:
 - se a temperatura for menor do que 20°C, então está frio
 - se a temperatura estiver entre 20°C e 30°C, então está agradável
 - se a temperatura for maior do que 30°C, então está quente

Tomada de Decisão

- Comando “if”:
 - comando básico para codificar tomada de decisão
 - se *expr* for verdadeira ($\neq 0$), executa o bloco de comandos 1
 - se *expr* for falsa ($= 0$), executa o bloco de comandos 2

```
if ( expr )  
{ bloco de comandos 1 }  
else  
{ bloco de comandos 2 }
```

ou

```
if ( expr )  
{ bloco de comandos }
```

Tomada de Decisão

```
/* temperatura (versao 1 - incorreta) */  
#include <stdio.h>  
  
int main (void)  
{  
    int temp;  
    printf("Digite a temperatura: ");  
    scanf("%d", &temp);  
    if (temp < 30)  
        if (temp > 20)  
            printf(" Temperatura agradável \n");  
    else  
        printf(" Temperatura quente \n");  
    return 0;  
}
```

Tomada de Decisão

```
/* temperatura (versao 1 - incorreta) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);
    if (temp < 30)
        if (temp > 20)
            printf(" Temperatura agradável \n");
        else
            printf(" Temperatura quente \n");
    return 0;
}
```

Tomada de Decisão

```
/* temperatura (versao 2) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf ( "Digite a temperatura: " );
    scanf ( "%d", &temp );
    if ( temp < 30 ) {
        if ( temp > 20 )
            printf ( " Temperatura agradável \n" );
    }
    else
        printf ( " Temperatura quente \n" );
    return 0;
}
```

```
/* temperatura (versao 3) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);

    if (temp < 10)
        printf("Temperatura muito fria \n");
    else if (temp < 20)
        printf(" Temperatura fria \n");
    else if (temp < 30)
        printf("Temperatura agradável \n");
    else
        printf("Temperatura quente \n");
    return 0;
}
```

```
/* temperatura (versao 3) */
#include <stdio.h>

int main (void)
{
    int temp;
    printf("Digite a temperatura: ");
    scanf("%d", &temp);

    if (temp < 10)
        printf("Temperatura muito fria \n");
    else if (temp < 20)
        printf(" Temperatura fria \n");
    else if (temp < 30)
        printf("Temperatura agradável \n");
        else printf("Temperatura quente \n");

    return 0;
}
```

Tomada de Decisão

- Estrutura de bloco:
 - declaração de variáveis:
 - só podem ocorrer no início do corpo da função ou de um bloco
 - (esta restrição não existe no C99)
 - escopo de uma variável:
 - uma variável declarada dentro de um bloco é válida no bloco
 - após o término do bloco, a variável deixa de existir

```
if ( n > 0 )  
    { int i; ... }  
...      /* a variável i não existe neste ponto do programa */
```

Tomada de Decisão

- Operador condicional:

- formato geral:

- se a condição for verdadeira, a expressão1 é avaliada;
caso contrário, a expressão2 é avaliada

```
condição ? expressão1 : expressão2;
```

- exemplo:

- comando

```
maximo = a > b ? a : b ;
```

- comando “if” equivalente

```
if ( a > b )  
    maximo = a;  
else maximo = b;
```

Construções com laços

- Exemplo:
 - fatorial de um número inteiro não negativo:

$$n! = n \times (n - 1) \times (n - 2) \dots 3 \times 2 \times 1$$

$$\text{onde } 0! = 1$$

Construções com laços

- Exemplo:
 - definição recursiva da função *fatorial*: $N \rightarrow N$
 $fatorial(0) = 1$
 $fatorial(n) = n \times fatorial(n-1)$
 - cálculo não recursivo de *fatorial*(*n*)
 - comece com:
 $k = 1$
 $f = 1$
 - faça enquanto $k \leq n$
 $f = f * k$
incremente k

Construções com laços

- Comando “while”:
 - enquanto *expr* for verdadeira, o bloco de comandos é executado
 - quando *expr* for falsa, o comando termina

```
while ( expr )  
{  
  bloco de comandos  
}
```

```

/* Fatorial */
#include <stdio.h>
int main (void)
{
    int k;
    int n;
    long int f = 1;
    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    k = 1;
    while (k <= n)
    {
        f = f * k;          /* a expressão "f = f * i" é equivalente a "f *= k"    */
        k = k + 1;        /* a expressão "k = k + 1" é equivalente a "k++"    */
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}

```

Construções com laços

- Exercício:
 1. Implemente o programa para calcular fatorial.
 2. Execute-o para 5, 10 e 20.
 3. Explique o comportamento do programa.

Construções com laços

- Comando “for”:
 - forma compacta para exprimir laços

```
for (expressão_inicial; expressão_booleana; expressão_de_incremento)  
{  
    bloco de comandos  
}
```

- equivalente a:

```
expressão_inicial;  
while ( expressão_booleana )  
{  
    bloco de comandos  
    ...  
    expressão_de_incremento  
}
```

```

/* Fatorial (versao 2) */
#include <stdio.h>

int main (void)
{
    int k;
    int n;
    int f = 1;

    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    for (k = 1; k <= n; k=k+1) { /* a expressão "k = k + 1" é equivalente a "i++" */
        f = f * k; /* a expressão "f = f * k" é equivalente a "f *= k" */
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}

```

```
/* Fatorial (versao 2) */
#include <stdio.h>

int main (void)
{
    int k;
    int n;
    int f = 1;

    printf("Digite um numero inteiro nao negativo:");
    scanf("%d", &n);

    /* calcula fatorial */
    for (k = 1; k <= n; k+1) {      /* o que acontece com este programa? */
        f = f * k;
    }
    printf(" Fatorial = %d \n", f);
    return 0;
}
```

Construções com laços

- Comando “do-while”:
 - teste de encerramento é avaliado no final

```
do  
{  
  bloco de comandos  
} while (expr);
```

```
/* Fatorial (versao 3) */
#include <stdio.h>
int main (void)
{
    int k;
    int n;
    int f = 1;
    /* requisita valor do usuário até um número não negativo ser informado */
    do
    { printf("Digite um valor inteiro nao negativo:");
      scanf ("%d", &n);
    } while (n<0);
    /* calcula fatorial */
    for (k = 1; k <= n; k++)
        f *= k;
    printf(" Fatorial = %d\n", f);
    return 0;
}
```

```
/* Fatorial (versao 4) */
#include <stdio.h>
int main (void)
{
    int k;
    int n;
    int f = 1;
    /* O que faz este programa? */
    do {
        printf("Digite um valor inteiro nao negativo:");
        scanf ("%d", &n);
        /* calcula fatorial */
        for (k = 1; k <= n; k++)
            f *= k;
        printf(" Fatorial = %d\n", f);
    } while (n>=0);
    return 0;
}
```

Construções com laços

- Interrupção de laços - Comando “break”:
 - termina a execução do comando de laço

```
#include <stdio.h>
int main (void)
{
    int i;
    for (i = 0; i < 10; i++) {
        if (i == 5)
            break;
        printf("%d ", i);
    }
    printf("fim\n");
    return 0;
}
```

A saída deste programa, se executado, será: 0 1 2 3 4 fim

Construções com laços

- Interrupção de laços - Comando “continue”:
 - termina a iteração corrente e passa para a próxima

```
#include <stdio.h>

int main (void)
{
    int i;
    for (i = 0; i < 10; i++ ) {
        if (i == 5) continue;
        printf("%d ", i);
    }
    printf("fim\n");
    return 0;
}
```

gera a saída: 0 1 2 3 4 ~~5~~ 6 7 8 9 fim

Construções com laços

- Interrupção de laços - Comando “continue”:
 - deve-se ter cuidado para não criar uma “iteração eterno”

```
/* INCORRETO */
#include <stdio.h>
int main (void)
{
    int i = 0;
    while (i < 10) {
        if (i == 5) continue;
        printf("%d ", i);
        i++;
    }
    printf("fim\n");
    return 0;
}
```

cria “iteração eterna” pois i não será mais incrementado quando chegar a 5

Construções com laços

- Comando “switch”:
 - seleciona uma entre vários casos
 (“op_k” deve ser um inteiro ou caractere)

```
switch ( expr )
{
  case op1: bloco de comandos 1; break;
  case op2: bloco de comandos 2; break;
  ...
  default: bloco de comandos default; break;
}
```

```

/* calculadora de quatro operações */
#include <stdio.h>
int main (void)
{
    float num1, num2;
    char op;
    printf("Digite: numero op numero\n");
    scanf ("%f %c %f", &num1, &op, &num2);
    switch (op)
    {
        case '+':      printf(" = %f\n", num1+num2); break;
        case '-':      printf(" = %f\n", num1-num2); break;
        case '*':      printf(" = %f\n", num1*num2); break;
        case '/':      printf(" = %f\n", num1/num2); break;
        default:       printf("Operador invalido!\n"); break;
    }
    return 0;
}

```

Resumo

```
if ( expr ) { bloco de comandos } else { bloco de comandos }
```

```
condição ? expressão1 : expressão2;
```

```
while ( expr ) { bloco de comandos }
```

```
for ( expr_inicial; expr_booleana; expr_de_incremento ) { bloco de comandos }
```

```
do { bloco de comandos } while ( expr );
```

```
switch ( expr ) {  
    case op1: bloco de comandos 1; break;  
    case op2: bloco de comandos 2; break;  
    ...  
    default: bloco de comandos default; break;  
}
```