Programação de Computadores II

Cap. 3 – Entrada e Saída e Controle de Fluxo

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

Slides adaptados dos originais dos profs.: Marco Antonio Casanova e Marcelo Gattass (PUC-Rio)

Tópicos

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

Capítulo 2 – I/O

Capítulo 3 – Controle de fluxo

- Entrada e saída: **`printf`** e **`scanf`**
- Tomada de decisão: **`if`** e **`?`**
- Construções com laços: **`while`**, **`for`**, **`do`**
- Seleção: **`switch`**
- Interruptores: **`break`** e **`continue`**

Entrada e Saída

- Função "printf":
  - possibilita a saída de valores segundo um determinado formato

```
printf (formato, lista de constantes/variáveis/expressões...);
```

```
printf ("%d %g", 33, 5.3);

tem como resultado a impressão da linha:
33 5.3
```

```
printf ("Inteiro = %d   Real = %g", 33, 5.3);

com saída:
Inteiro = 33   Real = 5.3
```

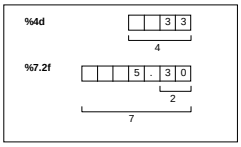

Entrada e Saída

- Especificação de formato:

%c	*especifica um char*
%d	*especifica um int*
%u	*especifica um unsigned int*
%f	*especifica um double (ou float)*
%e	*especifica um double (ou float) no formato científico*
%g	*especifica um double (ou float) no formato mais apropriado (%f ou %e)*
%s	*especifica uma cadeia de caracteres*

Entrada e Saída

- Especificação de tamanho de campo:



Entrada e Saída

- Impressão de texto:

```
printf("Curso de Estruturas de Dados\n");

exibe na tela a mensagem:
Curso de Estruturas de Dados
```

Entrada e Saída

- Função “scanf”:
  - captura valores fornecidos via teclado

```
scanf (formato, lista de endereços das variáveis...);
```

```
int n;
scanf ("%d", &n);

valor inteiro digitado pelo usuário é armazenado na variável n
```


Entrada e Saída

- Especificação de formato:

%c	*especifica um char*
%d	*especifica um int*
%u	*especifica um unsigned int*
%f,%e,%g	*especificam um float*
%lf, %le, %lg	*especificam um double*
%s	*especifica uma cadeia de caracteres*

Entrada e Saída

- Função “scanf” (cont.):
  - caracteres diferentes dos especificadores no formato servem para cercar a entrada
  - espaço em branco dentro do formato faz com que sejam "pulados" eventuais brancos da entrada
  - %d, %f, %e e %g automaticamente pulam os brancos que precederem os valores numéricos a serem capturados

```
scanf ("%d:%d", &h, &m);

valores (inteiros) fornecidos devem ser separados pelo
caractere dois pontos (:)
```


Tomada de Decisão

- Comando “if”:
  - comando básico para codificar tomada de decisão
    - se expr for verdadeira (≠ 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 }
```


Exemplo

```
/* nota */
#include <stdio.h>

int main (void)
{
   float nota ;
   printf("Digite sua nota: ");
   scanf("%f", &nota);
   if (nota >= 7 ){
      printf(" Boa nota, parabens! \n");
   }
   else  {
      printf(" Voce precisa melhorar. \n");
   }
   return 0;
}
```


Exemplo

```
/* nota */
#include <stdio.h>

int main (void)
{
   float nota ;
   printf("Digite sua nota: ");
   scanf("%f", &nota);
   if (nota >= 7 )
      printf(" Boa nota, parabens! \n");

   else
      printf(" Voce precisa melhorar. \n");

   return 0;
}
```

Bloco de comandos

```
{
 comando1;
 comando2;
 ...
 }
```

ou

```
 comando;
```


Tomada de Decisão

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

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;
}
```

Em C, um `else` está associado ao último `if` que não tiver seu próprio `else`.

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;
}
```

```c
/* 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;
}
```



```c
/* 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;
}
```



```c
/* 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

```c
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

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

  - exemplo:
    - comando

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

    - comando “if” equivalente

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


Construções com laços

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

$$n! = n \times (n-1) \times (n-2) ... 3 \times 2 \times 1$$
$$onde: 0! = 1$$

Construções com laços

- Exemplo:
  - definição recursiva da função *fatorial:* $\mathcal{N} \rightarrow \mathcal{N}$

    fatorial(0) = 1

    fatorial(n) = n x fatorial(n-1)

  - cálculo não recursivo de *fatorial(n)*
    - **comece com:**

      k = 1

      fatorial = 1
    - **faça enquanto** ***k ≤ n***

      fatorial = fatorial * 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 i;
   int n;
   long int f = 1;
   printf("Digite um numero inteiro nao negativo:");
   scanf("%d", &n);

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


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 i;
   int n;
   int f = 1;

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

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



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

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

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

  /* calcula fatorial */
  for (i = 1; i <= n; i+1) {      /* o que acontece com este programa?      */
    f = f * i;
  }
  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 i;
   int n;
   int f = 1;
   /* requisita valor até um número não negativo ser informado */
   do
   {
      printf("Digite um valor inteiro nao negativo:");
      scanf ("%d", &n);
   } while (n<0);
   /* calcula fatorial */
   for (i = 1; i <= n; i++)
      f *= i;
   printf(" Fatorial = %d\n", f);
   return 0;
}
```



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