

# Comandos de Controle: Tomada de Decisão e Laços ou Malhas de Repetição

Prof<sup>a</sup>.Dr<sup>a</sup>.Thatyana de Faria Piola Seraphim (ECO)  
Prof.Dr.Enzo Seraphim (ECO)

Universidade Federal de Itajubá

thatyana@unifei.edu.br  
seraphim@unifei.edu.br

# Comandos de Controle

## Bloco de Comandos

- ▶ **Blocos de comando:** são grupos de comandos que devem ser tratados como uma unidade lógica.
- ▶ O início de um bloco em C é marcado por um “{” e o término por um “}”
- ▶ O bloco de comando serve para criar um grupo de comandos que devem ser executados juntos.
- ▶ Usa-se o bloco de comandos quando se usa comandos de teste em que deve-se escolher entre executar dois blocos de comandos.
- ▶ Um bloco de comandos pode ser utilizado em qualquer trecho de programa que se pode usar um comando C.
- ▶ Os comandos de controle especificam a ordem em que a computação é feita no programa.

# Comandos de Controle

## Expressão Condicional

- ▶ Muitos comandos em C contam com uma **expressão condicional** que determina o curso da ação.
- ▶ Uma expressão condicional pode representar um valor **false** ou **true**.
- ▶ Um valor 0 é **false**.
- ▶ Um valor **true** é qualquer valor diferente de zero (incluindo números negativos).
- ▶ Para utilizar a constante **true** ou **false** deve-se definir o tipo *bool*.

### Definição de *bool*

```
1 → typedef enum {false, true} bool;
```

The image shows the C code for defining the `bool` type. The word `enum` is circled in red. Below the `false` and `true` literals, there are red circles containing the values 0 and 1, respectively, indicating their underlying integer representations. The word `bool` is also circled in red.

# Estruturas de Decisão



# Comandos de Controle

## Estruturas de Decisão: *if-else*

- ▶ O comando **if-else** é usado para expressar decisões, ou seja, é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.

### Sintaxe do **if**

```
1  if( expressao ){  
2      comando1;  
3  }//end if
```

*if((a==3) && (b==5))*  
Handwritten annotations: *a=3* (checked), *b==5* (checked), *&&* (checked), *ou* (or) below the *&&*.

*&& => E*  
*|| => ou*

### **if** com **else** opcional

```
1  if( expressao ){  
2      comando1;  
3  }//end if  
4  else{  
5      comando2;  
6  }//end else
```

*a=10;*  
*printf(" ");*  
*scanf(" ");*

# Comandos de Controle

## Estruturas de Decisão: *if-else*

### Sintaxe do *if*

```
1 if( expressao ){ // verdadeiro  
2     comando1; ← (true)  
3 } //end if  
4 else{ // falso  
5     comando2; ←  
6 } //end else
```

► expressão é avaliada:

- se for **true** (valor de expressão tem que diferente de zero), o comando ou o bloco que forma o corpo do **if** é executado;
- se for **false**, o comando ou o bloco que é o corpo do **else** é executado.

### Atenção

Apenas o código associado ao **if** ou o código associado ao **else** será executado, **nunca ambos**.

# Comandos de Controle

## Estruturas de Decisão: *if-else*

### Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      float nota;
4      printf("Digite uma nota: ");
5      scanf("%f", &nota);
6      if(nota >= 60){
7          printf("Aprovado");
8      }//end if
9      else{
10         printf("Reprovado");
11     }//end else
12     return 0;
13 }//end main
```

### Resultado

Digite uma nota: 75  
Aprovado

# Comandos de Controle

## Estruturas de Decisão: *ifs* Aninhados

- ▶ Quando for necessário escolher entre mais de uma opção pode-se utilizar ***if's*** aninhados.

### Sintaxe do *if*

```
1  if( expressao1 ){
2      comando1;
3  }//end if
4  else{
5      if( expressao2 ){
6          comando2;
7      }//end if
8      else{
9          comando3;
10     }//end else
11 }//end else
```



# Comandos de Controle

## Estruturas de Decisão: *ifs* Aninhados

### Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      float nota;
4      printf("Digite a nota do aluno = ");
5      scanf("%f", &nota); //digitar nota
6      if(nota >= 70){
7          printf("Aprovado Direto");
8      }//end if
9      else{
10         if(nota >= 50){
11             printf("Aprovado com Exame");
12         }//end if
13         else{
14             printf("Reprovado");
15         }//end else
16     }//end else
17     return 0;
18 }
```

# Comandos de Controle

## Comandos de Seleção: *switch...case*

- ▶ Para diversas comparações com uma mesma variável utilizamos o conjunto *switch...case*.
- ▶ A comparação é feita apenas com números e caracteres.
- ▶ As comparações são apenas de igualdade.
- ▶ Como boa prática de programação deve-se utilizar sempre um caso padrão **default**.

### Sintaxe do *switch...case*

```
1 switch( variavel ){  
2     case 1:  
3         comando1;  
4         break;  
5     case 2:  
6         comando2;  
7         break;  
8     default:  
9         comando3;  
10        break;  
11 } //end switch
```

# Comandos de Controle

Comandos de Seleção: *switch...case*

## Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      int tipo;
4      printf("Digite um tipo de combustivel = ");
5      scanf("%d", &tipo); //1 - alcool, 2 - gasolina
6      switch( tipo ){
7          case 1:
8              printf("Imposto = 4% do carro.");
9              break;
10         case 2:
11             printf("Imposto = 2% do carro.");
12             break;
13         default:
14             printf("Tipo de carro desconhecido!");
15             break;✗
16     }//end switch
17     return 0;
18 }//end main
```

# Laços ou Malhas de Repetição



# Laços ou Malhas de Repetição

## Introdução

- ▶ Todo loop **deve** possuir uma condição que indique quando este deve terminar. Uma condição mal feita pode prender o programa dentro do *loop*.
- ▶ Esta é uma das causas mais comuns para o "travamento" dos aplicativos, comumente chamada de **loop infinito**.

## Laços de repetição

Estruturas computacionais que permitem a repetição de um trecho de código N vezes ou enquanto uma condição for verdadeira.

# Laços ou Malhas de Repetição

## Repetição com teste no início - **while**

- ▶ A **repetição com teste no início do loop** é usada para repetir N vezes uma ou mais instruções.
- ▶ Não é necessário conhecer com antecedência o número de repetições.
- ▶ O controle do *loop* é feito através de uma condição.
- ▶ Para que o sistema **NÃO** entre em "*loop* infinito" esta condição **TEM** que ser alterada em algum momento **DENTRO** do *loop*.

### Sintaxe do **while**

```
1 while( condicao ){  
2     comando1;  
3     comando2;  
4     alteracao da condicao;  
5 } //end while
```

# Laços ou Malhas de Repetição

Repetição com teste no início - **while** - Atividade 1

Fazer um programa que:

- ▶ Leia o valor do salário dos funcionários de uma empresa.
- ▶ Ao terminar de ler os valores, deve imprimir a soma dos salários.
- ▶ A quantidade de funcionários não é conhecida.

# Laços ou Malhas de Repetição

## Repetição com teste no início do *loop* - Atividade 1

### Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      float total = 0.0, salario=1;0
4      while( salario > 0 ){ //0 - sai do programa
5          printf("Digite o valor de salario = ");
6          scanf("%f", &salario); ①
7          total = total + salario; ⇒ total += salario;
8      } //end while
9      printf("Somatorio = %f", 300total);
10     return 0;
11 } //end main
```

Handwritten notes:   
total = 300 (in a green box)   
salario = 0 (in an orange box)



# Laços ou Malhas de Repetição

Repetição com teste no final - **do...while**

- ▶ Assim como a instrução **while** a instrução **do..while** é utilizada para repetirmos um bloco do algoritmo diversas vezes.
- ▶ A diferença é o ponto onde a verificação da condição é realizada.
- ▶ Mesmo que a condição seja falsa desde o início, na estrutura **do..while** o bloco é executado pelo menos uma vez.

## Sintaxe do *do...while*

```
1  do{  
2      comando1;  
3      comando2;  
4      alteracao da condicao  
5  }while( condicao )
```

# Laços ou Malhas de Repetição

Repetição com teste no final - **do...while**

## ATENÇÃO NOVAMENTE

O controle do *loop* **TAMBÉM** é feito através de uma condição. Portanto, é necessário que essa condição seja alterada dentro do *loop*.

# Laços ou Malhas de Repetição

Repetição com teste no final - **do...while**- Atividade 1

Fazer um algoritmo que:

- ▶ Leia o valor do salário dos funcionários de uma empresa.
- ▶ Ao terminar de ler os valores deve imprimir a soma dos salários.
- ▶ A quantidade de funcionários não é conhecida.

# Laços ou Malhas de Repetição

Repetição com teste no final - **do...while**- Atividade 1

## Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      float total = 0.0, salario;
4      do{
5          printf("Digite o valor de salario = ");
6          scanf("%f", &salario);
7          100total = 100total + 0salario;
8      }while(salario > 0) //0 - sai do programa
9      printf("Somatorio = %f\n", total);
10     100return 0;
11 } //end main
```

total = 100

salario = 0

# Laços ou Malhas de Repetição

## Repetição com variável de controle - **for**

- ▶ Diferentemente das duas formas de *loop* apresentadas anteriormente a repetição com variável de controle **for**, é utilizada para repetir um bloco de instruções com uma quantidade de repetições pré-estabelecida.
- ▶ Para atingir este objetivo utilizamos dentro desta estrutura uma variável que trabalha como um contador. Esta indicará a quantidade de vezes que o bloco de instruções será repetido.

### Sintaxe do **for**

```
1 for(inicializacao; teste; incremento){  
2     comandos;  
3 }//end for
```

*for(i = 1; i < 5; i++) {*

*para i de 1 até 5 passo 1*  
*i <*

# Laços ou Malhas de Repetição

Repetição com variável de controle - **for** - Atividade 1

Fazer um programa em C que:

- ▶ Leia cinco valores dados pelo usuário.
- ▶ Some o triplo de cada valor.
- ▶ Imprima o somatório na tela.

# Laços ou Malhas de Repetição

Repetição com variável de controle - **for** - Atividade 1

## Exemplo

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      int total = 0, cont, num;
4      for(cont = 0; cont < 5; cont++){
5          printf("Digite um numero = ");
6          scanf("%d", &num);
7          total = total + (num * 3);
8      } //end for
9      printf("Somatorio = %d", total);
10     return 0;
11 } //end main
```

num: 1 cont: 4

total 27

0 1 2 3 4 5

# Laços ou Malhas de Repetição

Repetição com variável de controle - **for** - Atividade 1

## Exemplo: imprime a tabela ASCII

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      int i, j;
4      for(i = 0; i < 16; i++){
5          for(j = 1; j < 16; j++){
6              printf("[%d] = %c \t", i+j*16, i+j*16);
7          } //end for j
8          printf("\n");
9      } //end for i
10     return 0;
11 } //end main
```

#define PI 3.1416





# Laços ou Malhas de Repetição

## Repetição com variável de controle - **for** - Atividade 1

[16]=	[32]=	[48]=0	[64]=@	[80]=P	[96]=`	[112]=p	[128]=	[144]=	[160]=	[176]=*	[192]=À	[208]=Ð	[224]=à	[240]=ð
[17]=	[33]=!	[49]=1	[65]=A	[81]=Q	[97]=a	[113]=q	[129]=	[145]=	[161]=i	[177]=±	[193]=Á	[209]=Ñ	[225]=á	[241]=ñ
[18]=	[34]="	[50]=2	[66]=B	[82]=R	[98]=b	[114]=r	[130]=	[146]=	[162]=¢	[178]=²	[194]=Â	[210]=Ò	[226]=â	[242]=ò
[19]=	[35]=#	[51]=3	[67]=C	[83]=S	[99]=c	[115]=s	[131]=	[147]=	[163]=£	[179]=³	[195]=Ã	[211]=Ó	[227]=ã	[243]=ó
[20]=	[36]=\$	[52]=4	[68]=D	[84]=T	[100]=d	[116]=t	[132]=	[148]=	[164]=¤	[180]=´	[196]=Ä	[212]=Ô	[228]=ä	[244]=ô
[21]=	[37]=%	[53]=5	[69]=E	[85]=U	[101]=e	[117]=u	[133]=	[149]=	[165]=¥	[181]=µ	[197]=Å	[213]=Õ	[229]=å	[245]=õ
[22]=	[38]=&	[54]=6	[70]=F	[86]=V	[102]=f	[118]=v	[134]=	[150]=	[166]=¦	[182]=¶	[198]=Æ	[214]=Ö	[230]=æ	[246]=ö
[23]=	[39]='	[55]=7	[71]=G	[87]=W	[103]=g	[119]=w	[135]=	[151]=	[167]=§	[183]=·	[199]=Ç	[215]=×	[231]=ç	[247]=÷
[24]=	[40]={	[56]=8	[72]=H	[88]=X	[104]=h	[120]=x	[136]=	[152]=	[168]=¨	[184]=¸	[200]=È	[216]=Ø	[232]=è	[248]=ø
[25]=	[41]=}	[57]=9	[73]=I	[89]=Y	[105]=i	[121]=y	[137]=	[153]=	[169]=©	[185]=¹	[201]=É	[217]=Ù	[233]=é	[249]=ù
[26]=	[42]=*	[58]=:	[74]=J	[90]=Z	[106]=j	[122]=z	[138]=	[154]=	[170]=ª	[186]=º	[202]=Ê	[218]=Ú	[234]=ê	[250]=ú
[27]=	[43]=+	[59]=;	[75]=K	[91]=[	[107]=k	[123]=[	[139]=	[155]=	[171]=«	[187]=»	[203]=Ë	[219]=Û	[235]=ë	[251]=û
[28]=	[44]=,	[60]=<	[76]=L	[92]=\	[108]=l	[124]=\	[140]=	[156]=	[172]=¬	[188]=¼	[204]=Í	[220]=Ü	[236]=ì	[252]=ü
[29]=	[45]=-	[61]=:	[77]=M	[93]=]	[109]=m	[125]=]	[141]=	[157]=	[173]=	[189]=½	[205]=Î	[221]=Ý	[237]=í	[253]=ý
[30]=	[46]=.	[62]=>	[78]=N	[94]=^	[110]=n	[126]=~	[142]=	[158]=	[174]=®	[190]=¾	[206]=Ï	[222]=Þ	[238]=î	[254]=þ
[31]=	[47]=/	[63]=?	[79]=O	[95]=_	[111]=o	[127]=	[143]=	[159]=	[175]=¯	[191]=¸	[207]=I	[223]=ß	[239]=ï	[255]=ÿ

# Laços ou Malhas de Repetição

Comandos de desvio **break** e **continue**

- ▶ Os comandos **break** e **continue** permitem ao programador alterar o fluxo do programa dentro de um loop



www.shutterstock.com · 14848414

# Laços ou Malhas de Repetição

## Comandos de desvio **break** e **continue**

### **break**

- ▶ Ao utilizar o comando **break**, o loop é parado imediatamente, independente das condições.
- ▶ O programa tem sequência no primeiro comando depois do loop.
- ▶ Também é utilizado em conjunto com o comando **switch**.

### **continue**

- ▶ Ao utilizar o comando **continue**, a iteração atual do loop para de ser executada e o loop reinicia.
- ▶ Se for usada dentro de um loop do tipo **for** o bloco de incremento é executado.

# Laços ou Malhas de Repetição

Comandos de desvio **break** e **continue** - Atividade 1

Fazer um algoritmo que:

- ▶ Leia o valor do salário dos funcionários de uma empresa.
- ▶ Ao terminar de ler os valores deve imprimir a soma dos salários.
- ▶ A quantidade de funcionários não é conhecida.

# Laços ou Malhas de Repetição

## Comandos de desvio **break** e **continue** - Atividade 1

### Exemplo

```
1  #include <stdio.h>
2  typedef enum {false,true} bool;
3  int main(int argc, char *argv[]){
4      float total = 0.0, salario;
5      while(true){
6          printf("Digite o salario = ");
7          scanf("%f", &salario);
8          if(salario==0){
9              break; //se o salario = 0, sai do loop
10             }//end if
11             total = total + salario;
12         }//end while
13     printf("Total salario = %f", total);
14     return 0;
15 }
```

