Programação Estruturada Aula 6 - Comandos de controle

Prof. Luis Carlos Retondaro

Técnico em Telecomunicações
2^o Ano

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Campus Petrópolis

2017

Sumário

- A linguagem C possui diversos comandos de controle que podem ser divididos nos seguintes grupos:
  - Condicional (seleção): if e switch;
  - Repetição (laço): for, while e do while;
  - Desvio (salto): break, continue, goto e return;
  - Rótulo: case, default (discutidos no switch) e o label (discutido no goto);

- Muitos comandos em C utilizam um teste condicional que determina o curso da ação;
- Uma expressão condicional tem como resultado final um valor verdadeiro ou falso;
- Em C (ao contrário de muitas outras linguagens) um valor verdadeiro é qualquer valor diferente de zero, incluindo números negativos e um valor falso é zero;
- Esse método para verdadeiro e falso permite a implementação eficiente de uma ampla gama de rotinas.

Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

Bloco de comandos

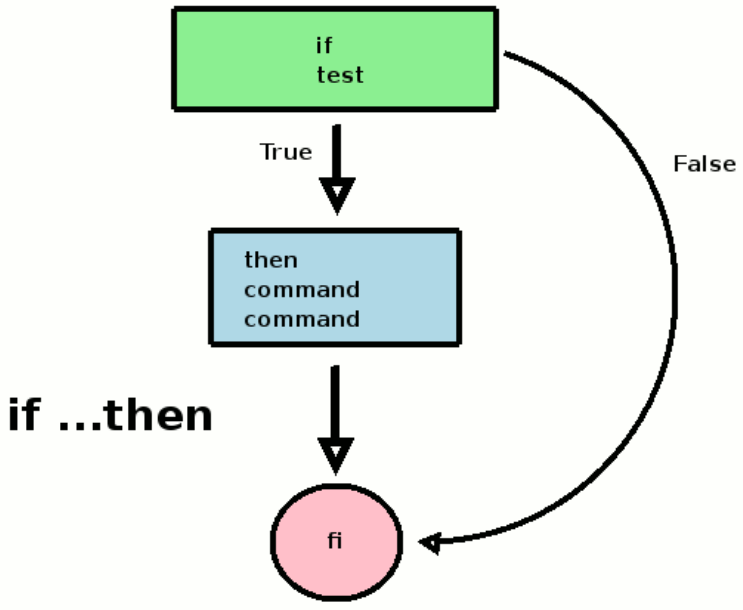
- É um conjunto de instruções agrupadas;
- Essas instruções são limitadas pelos caracteres { e }.

```
main(){
  int a;
  a=1;
  printf("Valor da variável a: %d\n", a);
}
```

O comando if $\cdots$ then

- Os comandos condicionais, permitem escolher o fluxo do programa a partir de uma ou mais alternativas;
- A seleção é baseada no valor de uma expressão de controle;
- O C suporta dois tipos de comandos de seleção (if e switch), além disso o operador ? é uma alternativa ao if em certas circunstâncias;
- O comando if $\cdots$ then executa uma seção do código se uma condicional for avaliada como true;

O comando if ··· then



O comando if ··· then

- Forma geral do comando condicional if...then:

```
if(expressão){
   comando(s);
}
```

- Onde comando pode ser uma única linha de instrução ou um bloco de instruções;

- As chaves de abertura { e fechamento } do bloco if ··· then são opcionais se a seção do código a ser executada no caso true tiver somente uma única instrução.

- **Exemplo:** programa que determina se um número é par.

```c
#include <stdio.h>

void main(){

   int num;
   printf("Digite um valor\n");
   scanf("%d",&num);

   if((num%2) !=0){
        printf("O número %d é ímpar\n",num);
   }
}
```

- Sabendo como o C representa valores verdadeiros ou falsos o programa pode ser modificado da forma:

```
#include <stdio.h>

void main(){

   int num;
   printf("Digite um valor\n");
   scanf("%d",&num);

   if(num % 2){
        printf("O número %d é ímpar\n",num);
   }
}
```

- **Exemplo:** programa que lê a nota do aluno e verifica se ele está aprovado.

```
//verifica aprovação do aluno

#include <stdio.h>

void main(){

   float notaAluno;
   printf("Digite a nota do aluno\n");
   scanf("%f",&notaAluno);

   //A cláusula "if": a nota deve ser maior que 60
   if(notaAluno > 60){
       // A cláusula "then": aprove o aluno
       printf("Aluno aprovado\n");
   }
}
```

- **Resultado da execução:**

```
//Execução 1

Digite a nota do aluno
55


//Execução 2

Digite a nota do aluno
77.89
Aluno aprovado
```

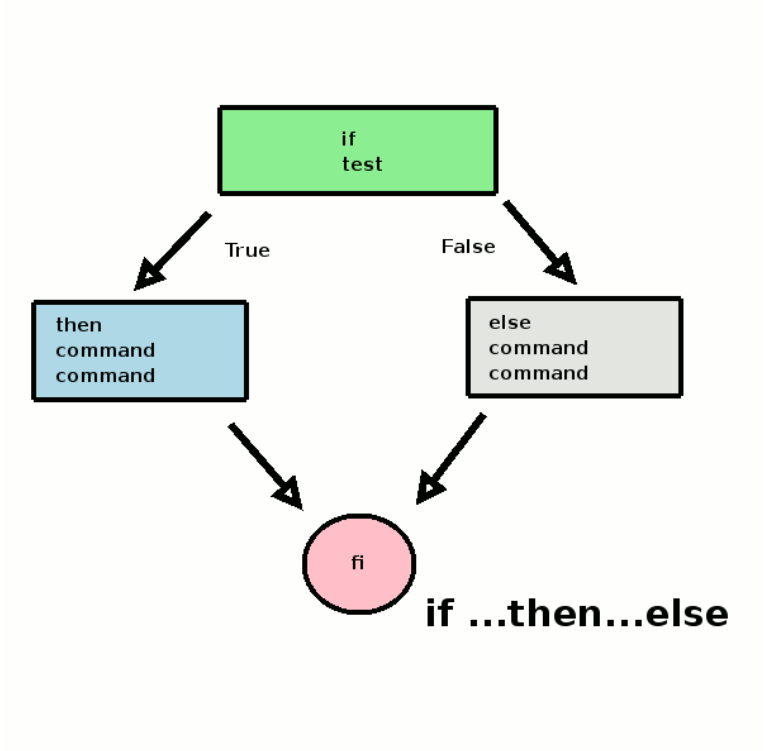
Exemplo: programa que lê duas notas e com a média verifica a aprovação do aluno.

```c
//Estrutura Condicional
#include<stdio.h>

    int main(){
    float nota1, nota2;
    float media;

    printf("Informe a nota da P1:\n");
    scanf("%f",&nota1);
    printf("Informe a nota da P2:\n");
    scanf("%f",&nota2);
    media = (nota1+nota2)/2;
    printf("Sua media é %.2f\n\n",media);

    //Aluno aprovado com a condição se a media é maior ou igual 6
    if (media >= 60)
      printf("Parabens! Voce esta aprovado!\n");
    return(0);
}
```

- **Resultado da execução:**

```
Informe a nota da P1:
50.3
Informe a nota 2:
70.5
Sua media é 60.40

Parabens! Voce esta aprovado!
```

- A instrução if ··· then ··· else permite especificar uma ação que será realizada quando a condição for verdadeira e uma ação diferente quando a condição for falsa.



Exemplo de uso do comando if... then ...else

```c
//Estrutura Condicional     Exemplo 2
#include<stdio.h>

int main(){
  float nota1, nota2;
  float media;

  printf("Informe a nota da P1:\n");
  scanf("%f",&nota1);
  printf("Informe a nota 2:\n");
  scanf("%f",&nota2);
  media = (nota1+nota2)/2;
  printf("Sua media é %.2f\n\n",media);
  //Aluno aprovado com a condição se a media é maior ou igual 6
  if (media >= 6)
      printf("Parabens! Voce esta aprovado!");
  else
      printf("Voce esta de prova final!\n");

  return(0);
}
```

- **Resultado da execução:**

```
Informe a nota da P1:
50.7
Informe a nota 2:
77.6
Sua media é 64.15

Parabens! Voce esta aprovado!
```

- Se a expressão é verdadeira o(s) comando(s) que forma(m) o corpo do if é(são) executado(s), caso contrário o bloco de comandos presente no corpo do else são executados;
- Nunca ambos (if e else) serão executados!!!
- O comando condicional que controla o if deve produzir um escalar (um número inteiro, um caractere ou um ponto flutuante);
- É raro usar ponto flutuante para controlar um comando condicional porque isso diminui consideravelmente a velocidade de execução.

- Uma coisa muito comum em programação é o teste de várias alternativas;
- Para isso podemos usar uma construção simples com ifs.

```
void main(){
 int ra;
 printf("Digite o RA do aluno\n");
 scanf("%d", &a);
 if(ra==10129)
  printf("Aluno João Carlos\n");
 if(ra==33860)
  printf("Aluna Aline Costa\n");
 if(ra==45693)
  printf("Aluno Cristiano Silva\n");
 if(...
  ...
}
```

- Uma construção comum é a aninhamento if-else-if:
- **Forma geral:**

```
if(Expressão)
  comando;
else
  if(expressão)
    comando;
  else
    if(Expressão)
      comando;
      .
      .
      .
      else
        comando;
```

- Todos os testes serão executados!!!
- Quando apenas uma de várias opções é verdadeira podemos usar a construção if $\cdots$ then $\cdots$ else.

```
void main(){
 int ra;
 printf("Digite o RA do aluno\n");
 scanf("%d", &a);
 if(ra==10129)
  printf("Aluno João Carlos\n");
 else if(ra==33860)
  printf("Aluna Aline Costa\n");
 else if(ra==45693)
  printf("Aluno Cristiano Silva\n");
 else if(...
  ...
 else
}
```

- Note que o if é um comando como outro qualquer, e como tal pode aparecer dentro do bloco de comandos de outro if;
- Um if aninhado é um comando if que é objeto de outro if ou else;
- Em C um comando else sempre se refere ao comando if mais próximo que está dentro do mesmo bloco do else e não está associado a outro if;
- **Exemplo:**

```
if(i){
   if(j)
     comando 1;
   if(k)
     comando 2;
   else
     comando 3;
} else
   comando 4;
```

- O padrão C ANSI especifica que pelo menos 15 níveis de aninhamento podem ser suportados;
- **Exemplo if aninhado:**

```
#include <stdio.h>
main(){
   char ch;
   printf("Digite uma letra entre a e z: ");
   ch = getchar();
   if( ch >= 'a')
     if( ch <= 'z')
       printf("\nVoce digitou um caracter!\n");
}
```

- O que acontece se o usuário digitar o caractere $'b'$?

- As condições são avaliadas de cima para baixo, assim que uma condição verdadeira é encontrada o comando associado a ela é executado e desvia do resto do aninhamento;
- Se nenhuma das condições for verdadeira, então o último else é executado, se o último else não está presente então nenhuma ação é executada.

```
#include <stdio.h>
main(){
   float num1, num2;
   char op;
   printf("\nDigite uma expressão no formato: num op num: ");
   scanf("%f %c %f", &num1, &op, &num2);
   if( op == '+' )
      printf(" = %f\n", num1+num2);
   else if( op == '-' )
      printf(" = %f\n", num1-num2);
   else if( op == '*' )
      printf(" = %f\n", num1*num2);
   else if( op == '/' )
      printf(" = %f\n", num1/num2);
   else
      printf("Operador inválido!!!\n");
}
```

- Quando o comando 2 é executado?

```
if(cond1
  if(cond)
    comando1;
else
  comando2;
```

```
if(cond1)
 if(cond2)
   comando1;
 else
   comando2;
```

```
if(cond1){
 if(cond2)
   comando1;
}else
   comando2;
```

- Use chaves e identação para deixar claro a qual bloco de comando(s) pertence cada comando!!!

Resumo

- Na construção de ifs aninhados quando uma condição é verdadeira, o bloco de comandos correspondente ao if "verdadeiro" será executado;
- Após a execução do bloco de comandos as outras alternativas não serão testadas;
- O último else (sem if) pode ser executado como uma opção padrão quando nenhuma das opções do if forem verdadeiras.

O operador ? alternativo

- Pode-se utilizar o operador ? em substituição ao comando if-else;
- O ? é denominado de operador ternário pois requer três operandos. Forma geral:

```
Expressão 1 ? Expressão 2 : Expressão 3;
```

- A expressão 1 é avaliada, se for verdadeira a Expressão 2 é executada sendo o resultado da expressão ? inteira. Se for falsa, então a expressão 3 é executada e se torna o valor da expressão.

```
x = 10;
y = x > 9 ? 100 : 200;

//de modo semelhante
x = 10;
if(x > 9)
   y = 100;
else
   y = 200;
```

Exemplo de uso do operador ?

```
#include<stdio.h>
void main(){
  int quad, i;

  printf("Digite um valor inteiro\n");
  scanf("%d",&i);
  quad = i > 0 ? i*i : -(i*i);
  printf("%d ao quadrado é %d\n", i, quad);
}
```

- O uso do operador ? não é restrito a atribuições, pode-se usar qualquer expressão ou função que não retorne um void.

- Nos comando condicionais pode-se usar qualquer expressão válida em C, isso significa que o programador não fica restrito às expressões envolvendo os operadores relacionais e lógicos;
- O programa precisa simplesmente chegar a um valor zero ou não zero.

```
#include <stdio.h>
void main(){
    int a, b;
    printf("Digite dois números\n");
    scanf("%d%d",&a,&b);
    if(b)
      printf("%d\n",a/b);
    else
      printf("Não existe divisão por zero!!!\n");
}
```

O comando switch

- O comando switch é um comando condicional de seleção múltipla que testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou caracter;
- Quando o valor coincide, os comandos associados àquela constante são executados. Forma geral:

```
switch(Expressão){
  case constante1:
       sequência de  comandos;
       break;
  case constante2:
       sequência de  comandos;
       break;
  .
  .
  .
  default:
       sequência de  comandos;
}
```

O comando switch

```c
#include<stdio.h>
void main(){
  char c;

  printf("1. Chegar ortografia\n");
  printf("2. Corrigir erros de ortografia\n");
  printf("3. Mostrar erros de ortografia\n");
  printf("Pressione qualquer outra tecla para abandonar\n");
  printf("\n\n *** Entre com a opção desejada\n");
  c = getchar();
  switch(c){
    case '1':
      checar_ortografia();
      break;
    case '2':
      corrigir_erros();
      break;
    case '3':
      mostrar_erros();
      break;
    default:
      printf("Nenhuma opção selecionada!\n");
  }
}
```

- O valor da expressão é testado, em ordem, contra os valores das constantes especificadas nos comandos case;
- Quando uma conincidência for encontrada, a sequência de comandos associada àquele case será executada até que o comando break ou o fim do comando switch seja alcançado;
- O comando default é executado se nenhuma coincidência for encontrada;
- O comando default é opcional, se não estiver presente, nenhuma ação será realizada se todos os testes falharem.

- O padrão ANSI C especifica que um comando switch pode ter pelo menos 257 comandos case, na prática o programador utilizará uma quantidade menor para obter mais eficiência;
- Embora case seja um rótulo, ele não existe sozinho fora de um switch;
- O comando break é um dos comandos de desvio em C e pode ser usado em comandos de repetição, etc;
- Quando um comando break é encontrado dentro de um switch a execução do programa salta para a linha de código seguinte ao comando switch.

Atenção!!!

- O comando switch difere do comando if porque o switch só pode testar igualdade, enquanto if pode avaliar uma expressão lógica ou relacional;
- Duas constantes case no mesmo switch não podem ter valores idênticos;
- Um comando switch dentro de outro comando switch podem ter valores de constantes case idênticos;
- Se constantes de caracteres são utilizadas em um comando switch elas são automaticamente convertidas para seus valores inteiros.

- Tecnicamente, os comandos break dentro do switch, são opcionais. Eles terminam a sequência de comandos associado a cada constante;
- Se o comando break é omitido a execução continua pelos próximos comandos case até que um break ou fim do switch seja alcançado;
- Um comando switch dentro de outro comando switch podem ter valores de constantes case idênticos;
- Se constantes de caracteres são utilizadas em um comando switch elas são automaticamente convertidas para seus valores inteiros.

O comando switch

```c
#include<stdio.h>
// Calcula o número de dias de um determinado mês.
void main(){
        int mes, ano, numDias = 0;

        printf("Digite o mes\n");
        scanf("%d",&mes);
        printf("Digite o ano\n");
        scanf("%d",&ano);
        switch(mes){
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                numDias = 31;
                break;
            case 4: case 6: case 9: case 11:
                numDias = 30;
                break;
            case 2:
                if (((ano % 4 == 0) && !(ano % 100 == 0)) || (ano % 400 == 0))
                    numDias = 29;
                else
                    numDias = 28;
                break;
            default:
                printf("Mês Inválido.\n");
        }
        printf("Número de Dias = %d \n", numDias);
}
```

- Pode-se ter comandos case sem ter comandos associados a ele, quando isso ocorre a execução passa para o case seguinte;
- Se nenhum comando break estiver presente a execução continua no próximo case;
- Os comandos associados aos cases não são blocos de código e sim sequência de código.

O comando switch aninhado

- Pode-se ter um switch como parte de sequência de comandos de um outro switch;
- Mesmo se as constantes dos cases do switch interno e externo tiverem valores comuns não haverá conflito.

```
switch(x){
  case 1:
  switch(y){
    case 0:
      printf("Erro de divisão por zero\n");
      break;
   case 1:
      divida(x,y);
  }
  case 2:
  .
  .
  .
}
```

- Os comandos de repetição permitem que um conjunto de instruções seja executado até que ocorra uma certa condição de parada;
- Na linguagem C existem 3 comandos de repetição: for, while, do-while;
- A condição de parada pode ser pré-definida como o que ocorre nos laços for, ou com o final em aberto como ocorre nos laços while e do while.

O laço for

- O comando for permite que um certo trecho de programa seja executado um determinado número de vezes, de maneira que se possa ter um bom controle sobre o loop;
- Sua forma geral:

```
for(inicialização; condição; incremento){
    lista de comandos;
    .
    .
    .
}
```

- O for permite muitas variações, entretanto geralmente:
  1. A inicialização é um comando de atribuição usado para colocar um valor inicial na variável de controle;
  2. A condição é uma expressão relacional que determina quando o laço acaba;
  3. O incremento define como a variável de controle varia cada vez que o laço é repetido.
- Esses três comandos do for devem ser separados por ";" ;
- Quando a condição do for se torna falsa a execução do programa continua no comando seguinte ao for.

- Sequência de funcionamento do comando for:
  1. Executa os comandos de inicialização;
  2. Testa a condição;
  3. Se a condição for falsa então executa o comando que está logo após o bloco subordinado ao for;
  4. Se condição for verdadeira então executa os comandos que estão subordinados ao for;
  5. Executa os comandos de incremento/decremento.

Exemplo: Impressão dos números de 1 a 100 na tela.

```c
// Imprime os números de 1 a 100.
    #include <stdio.h>
    void main(){

        int cont;

        for(cont = 1; cont <=100; cont++)
            printf("%d  ",cont);

        printf("\n");
    }
```

Exemplo: Impressão dos números de 1 a 100 na tela Resultado da execução.

```
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22
23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
43 44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100
```

Exemplo: for com múltiplos comandos.

```
#include <stdio.h>
void main(){

   int z;
   for(i = 100; i !=65; i-=5){
        z = i*i;
        printf("O quadrado de %d:, %d", x, z);
     }
}
```

Nos laços for o teste condicional sempre é executado no inicio do loop, então dependendo da inicialização e da condição de parada o código presente dentro do for pode não ser executado nem uma única vez.

Mais de uma variável controlando o for

- O C permite que o programador utilize mais de uma variável para controlar o for, separando essas variáveis por vírgula. Exemplo;

```
...

for(x=0,y=0; x+y<10; ++x)
  y = 2*x;

...
```

- A vírgula separa os dois comandos de inicialização;
- As duas variáveis controlam o laço.

funções como controle do for

- cada uma das três seções do for pode consistir em qualquer expressão válida em C;

```
#include<stdio.h>

int printmsg(){
 printf("\ndigite um número\n");
 return 0;
}
int readnum(){
 int t;
 scanf("%d", &t);
 return t;
}
int sqrnum(int num){
 printf("%d", num*num);
 return num*num;
}
void main(){
 int t;
 for(printmsg(); t = readnum(); printmsg())
     sqrnum(t);
}
```

Laço infinito

- O for pode ser utilizado para criar um laço infinito, dado que nenhuma expressão do for é obrigatória.
- **Exemplo:**

```
...
int x = 0;
for(;;){
   printf("x = %d\n",x);
   x++;
}
...
```

Laço infinito

- O laço for sem nenhuma expressão, só não é infinito se tiver um comando break dentro dele, assim a execução continua no código seguinte ao for.

```
...
char ch='\0';

for(;;){
  ch = getchar(); // obtem um caracter
  if(ch == 'A')
     break; // sai do for
}

printf("Foi digitado um A\n");

...
```

- O comando for deve ser usado sempre que:
  - Soubermos exatamente quantas vezes o laço deve ser repetido;
  - O teste deva ser feito antes da execução de um bloco de comandos;
  - Houver casos em que o laço não deva ser repetido nenhuma vez.

Observações sobre o comando for

- Os comandos de inicialização são executados apenas 1 vez;
- O contador é incrementado/decrementado sempre ao final da execução do bloco;
- O teste é feito sempre antes do início da execução do bloco de comandos.

O laço while

- Outro comando de repetição disponível em C é o laço while;
- O comando de repetição while tem duas partes: a condição de parada (teste) e o corpo da repetição;
- Forma geral:

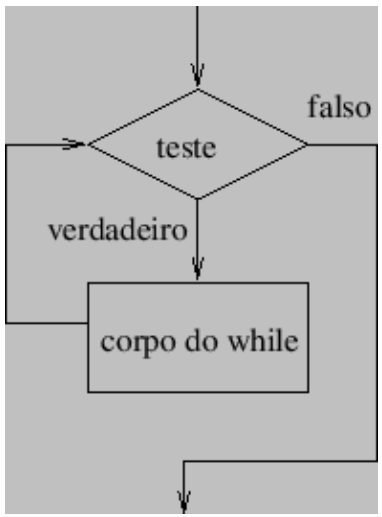
```
while(condição){
  lista de comandos;
}
```

O laço while

```
while(condição){
  lista de comandos;
}
```

- Onde comando pode ser um comando vazio, um comando simples ou um bloco de comandos;
- A condição pode ser qualquer expressão válida em C;
- Os comandos dentro do while serão repetidos enquanto a condição for verdadeira, quando a condição for falsa o controle do programa passa para a linha de código após o while.

- A condição é inicialmente avaliada para verificar se o laço deve terminar;
- Se a condição for verdadeira, o corpo da repetição é executado;
- Depois desta execução, o processo é repetido a partir da expressão teste.



- **Exemplo 1: Função que espera a entrada 'A'**

```
#include <stdio.h>

void main(){
  char ch;
  ch = '\0';

  while(ch !='A')
    ch = getchar();
  printf("Entrada: %c\n",ch);
 }
```

- Primeiro ch é inicializado com nulo. Então o laço verifica a igualdade, como inicialmente ch $\neq$ 'A' o laço começa;

- Toda vez que o usuário pressiona uma tecla o teste é realizado novamente, ao digitar 'A' a condição se torna falsa e o while termina.

- Como no for, o while também verifica a condição no início do laço;
- Também não é necessário existir nenhum comando no corpo do while;

```
char ch;

while(ch = getchar() != 'A');
```

- O loop será executado até que o usuário digite um caractere 'A'.

- Se o corpo do while possuir mais de uma linha de comando que devem ser repetidas é necessário o uso do { e };
- **Exemplo 2:**

```
#include <stdio.h>

main( ){

  int total = 0, num;

  while( total < 20) {
    printf("Total = %d\n", total);
    printf("\nEntre com um numero: \n");
    scanf("%d", &num);
    total += num;
  }

  printf("\nTotal final = %d\n", total);
}
```

- **Exemplo 2: Resultado da execução**

```
Total = 0

Entre com um numero:
1
Total = 1

Entre com um numero:
5
Total = 6

Entre com um numero:
0
Total = 6

Entre com um numero:
10
Total = 16

Entre com um numero:
15

Total final = 31
```

Considerações

- O comando while deve ser usado sempre que:
  - Não soubermos exatamente quantas vezes o laço deve ser repetido;
  - Quando o teste deve ser feito antes de iniciar a execução de um bloco de comandos;
  - Houver casos em que o laço não deva ser repetido nenhuma vez.
- Tenha atenção especial com o teste do comando while. As variáveis usadas no teste devem ter sido inicializadas antes do teste.

O laço do-while

- O comando do-while permite que um certo trecho de programa seja executado ENQUANTO uma certa condição for verdadeira;
- Ao contrário dos comandos de repetição for e while o do-while verifica a condição de parada no final do loop;
- Isso significa que o laço do-while sempre será executado pelo menos uma vez;
- Forma geral:

```
do{
  lista de comandos;
}while(condição);
```

- O laço do while repete até que a condição se torne falsa.

- Funcionamento do comando do-while:
  - Executa os comando dentro do bloco do-while;
  - Testa a condição;
  - Se a condição for falsa então executa o comando que está logo após o bloco subordinado ao do-while;
  - Se condição for verdadeira então volta ao passo 1.

- **Exemplo:** Programa que lê números do teclado até que encontre um número menor ou igual a 100.

```
#include <stdio.h>

void main(){
int num;

do{
    printf("\nDigite um valor\n");
    scanf("%d", &num);

}while(num > 100);
}
```

- **Exemplo:** Resultado da execução

```
Digite um valor
235

Digite um valor
632

Digite um valor
9523

Digite um valor
83
```

- Talvez o uso mais comum do comando do-while seja em uma rotina de seleção por menu;
- Quando o usuário entra com uma resposta válida, a opção escolhida é executada;
- Respostas inválidas causam uma repetição do laço.

```
#include<stdio.h>

void imprimeMenu(){
    printf("1. Verificar ortografia\n");
    printf("2. Corrigir erros de ortografia\n");
    printf("3. Mostrar erros de ortografia\n");
    printf(" Digite sua escolha \n");

    //continua...
}
```

O laço do-while

```
//continuacao...

void main(){
  char c;

  do{
    imprimeMenu();
    c = getchar();
    if(c == '\n')
      c = getchar();
    switch(c){
      case '1':
        verificar();
        printf("Ortografia verificada\n");
        break;
      case '2':
        corrigir();
        printf("erros corrigidos\n");
        break;
      case '3':
        mostrar();
        printf("Erros mostrados\n");
        break;
      default:
        printf("Opção inválida!\n");
    }
  }while(c != '1' && c != '2' && c != '3');
}
```

- O comando do-while deve ser usado sempre que:
  - Não soubermos exatamente quantas vezes o laço deve ser repetido;
  - O teste precisa ser feito depois da execução de um bloco de comandos;
  - O bloco de comandos deve se executado pelo menos 1 vez.

- É possível colocar um laço dentro do outro.

- **Exemplo:** laço aninhado usando while (impressão no formato de matrizes)

```
#include <stdio.h>

int main( ){

  int linha, coluna;

  linha = 1;
  while (linha < 5){
    coluna = 1;
    while (coluna < 5){
      printf( "%3d", linha * coluna );
      coluna += 1;
    }
    printf("\n");
    linha += 1;
  }
  printf( "\n" );
}
```

- **Exemplo:** Resultado da execução.

```
 1  2  3  4
 2  4  6  8
 3  6  9 12
 4  8 12 16
```

- C tem cinco comandos que realizam um desvio incondicional: return, goto, break, exit e continue;
- O return, goto e o break podem ser usados em qualquer lugar do programa;
- Os comandos break e continue podem ser usados com qualquer dos comandos de repetição, sendo que o break também pode ser usado com o switch.

- É usado para retornar o controle de uma função com um valor ou não;
- Ele é considerado um comando de desvio porque faz com que a execução retorne ao ponto em que a chamada da função foi feita;
- Se return tem um valor associado a ele, este é o valor de retorno da função. Se a função tem um retorno e nenhum foi especificado então um lixo é retornado;
- Forma geral:

```
return expressão;
```

- Pode-se usar mais de um comando return dentro de uma mesma função, porém a função pára de executar tão logo encontre o primeiro return;

- **Atenção:** Uma função declarada como void não pode ter um comando return que especifique um valor.

```
float calculaMedia(int vet[], int n){
   int i;
   float media = 0.0;

   for(i=0; i<n; i++)
     media += vet[i];

   media /= n;
   return(media);
}
```

- É um comando que desvia a execução para qualquer parte do programa;
- Dado que o C tem um conjunto de estruturas de controle abrangente e ainda permite um controle adicional usando break e continue há pouca necessidade do goto;
- O goto deixa o programa menos legível e não há nenhuma situação onde o goto seja necessário, por esses motivos ele não é muito utilizado pelo programadores.
- Forma geral:

```
goto rotulo;
.
.
.
rotulo:
```

- O goto requer um rótulo para sua operação;
- Um rótulo é um identificador em C seguido por um dois pontos (:);
- O rótulo precisa estar em dois lugares do programa: No lugar para onde a execução deve saltar e após o comando goto;
- Comandos de repetição podem ser criados usando o comando de desvio goto.

```
...
x = 1
loop:
x++;
printf("valor de x: %d  ",x);
if(x < 100)
 goto loop;
...
```

- O comando break faz com que a execução de um laço seja terminada, passando a execução para o próximo comando depois do final do laço;
- O comando break tem duas formas de ser usado:
  - Para terminar um case em um switch (já visto!);
  - Para forçar uma terminação imediata de um laço;
- Quando o comando break é usado dentro de um laço, a repetição é imediatamente encerrada e o controle do programa segue após o laço.

- **Exemplo:**

```c
#include <stdio.h>

void main(){
  int t, i;

  for(i=0; i<100; i++){
    printf("%d", i);
    if(i%13 == 0)
      break;
    }
}
```

- O comando break é usado em laço quando uma condição especial deve provocar um término imediato do loop.

- Um comando break provoca a saída do loop mais interno;
- **Exemplo:**

```c
#include <stdio.h>

void main(){
  int t, count, i, j;

  for(i=0; i<100; i++){
    count = 1;
    for(j=i; j<100;j++){
      printf("%d ",count);
      count++;
      if(j%count==0)
        break;
    }
  }
}
```

- Assim também um break dentro do switch só afetará o switch e não qualquer comando em que ele estiver envolvido.

- Os dois programas a seguir são equivalentes:

```c
#include <stdio.h>

void main(){
  int i;

  for(i=1; ; i++){
    if(i > 10)
      break;
    printf("%d \n",i);
  }
}
```

```c
#include <stdio.h>

void main(){
  int i;

  for(i=1; i<=10; i++){
    printf("%d \n",i);
  }
}
```

Função exit()

- Da mesma forma que se pode sair de um laço também é possível sair de um programa usando a função exit();
- A função exit() provoca um término imediato do programa inteiro, forçando um retorno ao sistema operacional;
- Forma geral:

```
void exit(codigo_de_retorno);
```

- O valor do código de retorno é enviado ao processo chamador (normalmente o SO), zero indica o término normal do programa.

- **Exemplo:**

```
...
  do{
    c = getchar();
    if(c == '\n')
      c = getchar();
    switch(c){
      case '1':
      verificar();
      printf("Ortografia verificada\n");
      break;
      case '2':
      corrigir();
      printf("erros corrigidos\n");
      break;
      case '3':
      mostrar();
      printf("Erros mostrados\n");
      break;
      case '4':
      printf("Abandonando o programa!\n");
      exit(0);
    }
  }while(c != '1' && c != '2' && c != '3');
}
```

- O comando continue faz com que a execução de um laço seja alterada para o final do mesmo;

```c
#include <stdio.h>

void main(){
  int i;
  for(i = 1; i<= 10 ; i++){
    if(i == 5)
      continue;
    printf("%d\n",i);
  }
  printf("Terminou o laço \n");
```

- O que será impresso?

- Esse comando trabalha de uma forma um pouco parecida com o break, porém ao invés de forçar a terminação ele força seguir para a próxima iteração;
- Para o comando for o continue faz com que o incremento e o teste sejam executados;
- Para os comandos while e o do while, o controle do comando passa para o teste condicional.

- **Exemplo:**

```c
//conta espaços
#include <stdio.h>
#include <string.h>

void main(){
  char frase[100];
  int i, espacos=0, tam;

  printf("Digite uma string: \n");
  gets(frase);
  tam = strlen(frase);
  for(i=0; i<tam; i++){
    if(frase[i] != ' ')
      continue;
    espacos++;
  }
  printf("%d espacos\n",espacos);
}
```

- O comando continue é utilizado em situações onde comandos dentro do laço só serão executados caso alguma condição seja satisfeita;

- Imprimindo a área de um círculo, mas apenas se o raio for par (e entre 1 e 10);

```
#include <stdio.h>

void main(){
  int r;
  double area;
  for(r = 1; r<= 10; r++){
    if((r%2) != 0)
      continue;
    area = 3.1415*r*r
    printf("%f\n",area);
  }
}
```

- Imprimindo a área de um círculo, mas apenas se o raio for par (e entre 1 e 10);
- Note que poderíamos escrever algo mais simples!

```c
#include <stdio.h>

void main(){
  int r;
  double area;
  for(r = 2; r<= 10; r+=2){
    area = 3.1415*r*r
    printf("%f\n",area);
  }
}
```

1. C Completo e Total, Herbert Schidt; Pearson Makron Books; 3a. Ed., 1997.

2. Linguagem C. DAMAS, Luis. 10a. Edição. LTC, 2014.