

Algoritmos e Programação – Parte 02
Instruções Condicionais
Estruturas de Controle de Fluxo e Repetição



Algoritmos e Programação

Instrução condicional

Sintaxe

```
if(<condição> ){  
    <instrução>  
}
```

```
if(<condição> )  
{  
    <instrução 1>  
    ...  
    <instrução N>  
}  
else {  
    <instrução A>  
    ...  
    <instrução Z>  
}
```

Algoritmos e Programação

Instrução condicional

```
#include <stdio.h>
int main () {
int a, b;
a=1;
b=2;
if (a>b) {
printf ("a maior que b");
}
else {
printf ("b maior que a");
}
getchar();
return (0);
}
```

Algoritmos e Programação

Instrução condicional

Os códigos funcionam? Existe diferença?

```
int varNumero;
printf ("Digite um numero: ");
scanf ("%d", &varNumero);

if (varNumero > 10){
    printf ("Numero MAIOR que 10");
}
else if (varNumero == 10){
    printf ("Voce digitou: 10");
}
else if (varNumero < 10){
    printf ("Numero MENOR que 10");
}
```

```
int varNumero;
printf ("Digite um numero: ");
scanf ("%d", &varNumero);

if (varNumero > 10){
    printf ("Numero MAIOR que 10");
}
if (varNumero == 10){
    printf ("Voce digitou: 10");
}
if (varNumero < 10){
    printf ("Numero MENOR que 10");
}
```

Algoritmos e Programação

Instrução condicional - operador ?

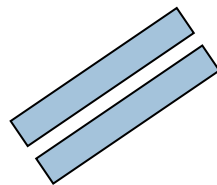
```
if (condição){  
    instrução 1;  
}  
else  
{  
    instrução 2;  
}
```

É equivalente a:

```
condição?instrução1:instrução2;
```

Algoritmos e Programação

Instrução condicional - operador ?



```
#include <stdio.h>

int main () {
float nota;
nota=9,5;
nota>=7?printf("Aprovado"):printf("Reprovado");

getchar();
return (0); }
```

```
#include <stdio.h>
int main () {
float nota;
nota = 9.5;
if (nota >= 7){
printf("Aprovado.");
}else{
printf("REPROVADO!"); }
getchar();
return (0);}
```

Algoritmos e Programação

Exemplo – O programa a seguir recebe como entrada dois números inteiros e gera como saída o resultado da divisão entre eles.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n1, n2;
    float res;
    printf ("Digite o dividendo inteiro: ");
    scanf ("%d", &n1);
    printf ("Digite o divisor inteiro: ");
    scanf ("%d", &n2);
    if (n2==0)
        printf ("Impossivel dividir!");
    else {
        res = (float) n1 / n2;
        printf ("Resultado da divisao: %.2f", res); }
    getch();
}
```

Algoritmos e Programação

Estruturas de controle de fluxo – switch/case

- Próprio para se **testar** uma **variável** em relação a **valores pré-estabelecidos**.
- Testa o conteúdo da variável e **executa a instrução** correspondente ao **case**;
- **break**, faz com que o switch seja **interrompido**;
- **default** é opcional;
- Não aceita expressões.

```
switch (variável)
{
    case constante_1:
        instrução 1;
        break;
    case constante_2:
        instrução 2;
        break;
    ...
    default _padrão;
}
```


Algoritmos e Programação

Estruturas de controle de fluxo – switch/case

```
switch (varNumero)
{
    case 9:
        printf ("O numero e igual a 9.");
        break;
    case 10:
        printf ("O numero e igual a 10.");
        break;
    default:
        printf ("O numero nao e nem 9 nem 10.");
}
```

Algoritmos e Programação

Estruturas de controle de fluxo – switch/case

```
//Exibir os dias da semana
#include <stdio.h>
#include <conio.h>
int main() {
    int num;
    printf ("Digite um numero de 1 a 7\n");
    scanf ("%d",&num);
    switch(num) {
    case 1:
        printf("Hoje he Domingo");
        break;
    case 2:
        printf("Hoje he Segunda");
        break;
    case 3:
        printf("Hoje he Terca");
        break;
```

```
case 4:
    printf("Hoje he Quarta");
    break;
case 5:
    printf("Hoje he Quinta");
    break;
case 6:
    printf("Hoje he Sexta");
    break;
case 7:
    printf("Hoje he Sabado");
    break;
default:
    printf("\n Numero Invalido"); }
getch();
return 0;
}
```

Algoritmos e Programação



- Exercícios:
- 1-Peça três números e mostre o maior entre eles.
- 2-Peça uma letra e mostre se ela é vogal ou consoante.

Algoritmos e Programação

1-Peça três números e mostre o maior entre eles.

```
#include <stdio.h>
#include <conio.h>
main() {
    int n1, n2, n3;
    printf("Digite tres numeros diferentes \n");
    scanf("%d" "%d" "%d",&n1, &n2, &n3);
    if (n1>n2) {
        if (n1>n3) {
            printf ("\nO maior eh: %d", n1);}
        else {
            printf ("\nO maior eh: %d", n3);}
    }
    else {
        if (n2>n3) {
            printf ("\nO maior eh: %d", n2);}
        else {
            printf ("\nO maior eh: %d", n3);}
    }
    getch();
}
```

Algoritmos e Programação

2-Peça uma letra e mostre se ela é vogal ou consoante.

```
#include <stdio.h>
#include <conio.h>
main() {
    char letra;
    printf ("Digite uma letra\n");
    scanf ("%c",&letra);
    switch(letra) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        printf("Eh uma vogal");
        break;
```

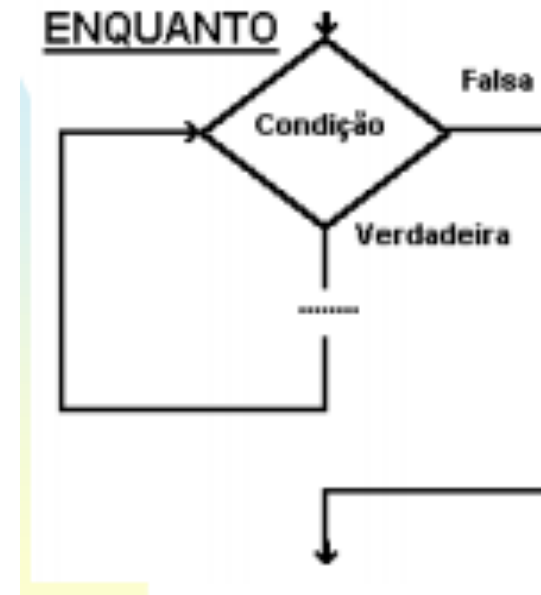
```
    case 'b':
    case 'c':
    case 'd':
    ...
    case 't':
    case 'w':
    case 'x':
    case 'y':
    case 'z':
        printf("Eh uma consoante");
        break;
    default:
        printf("\n Letra Invalida"); }
    getch();
}
```

Algoritmos e Programação

Loops de Repetição – **while**

Sintaxe

```
while( <condição> )  
{  
    <instrução 1>  
    ...  
    <instrução n> }  
}
```



Algoritmos e Programação

Loops de Repetição – **while**

- Repete enquanto a condição for verdadeira.
- Conhecido como laço controlado logicamente por uma expressão booleana.
- A condição é testada:
 - Antes da primeira execução do bloco.
 - E após a cada repetição.
- while com apenas um comando no corpo não precisa de chaves.
- `while (x != 10) scanf ("%d", &x);`

Algoritmos e Programação

Loops de Repetição – **while**

Exemplo

```
#include <stdio.h>

int main ()
{
    int numero;

    printf("Digite um numero: ");
    printf("\nDigite '0' para finalizar.\n\n");
    while (numero != 0 ){
        scanf("%d", &numero);
        printf("\n Voce digitou: %d \n Digite um novo numero: ", numero);
    }
    getchar();
    return(0);
}
```


Algoritmos e Programação

Loops de Repetição – **do...while**

Sintaxe

```
do{  
    <instrução 1>  
    ....  
    <instrução n>  
}while(<condição>);
```



Algoritmos e Programação

```
#include <stdio.h>
int main ()
{
    int i;
    do
    {
        printf ("\n\nEscolha a fruta pelo numero:\n\n");
        printf ("\t(1)...Mamao\n");
        printf ("\t(2)...Abacaxi\n");
        printf ("\t(3)...Laranja\n\n");
        scanf("%d", &i);
    } while ((i<1)||i>3));
    switch (i)
    {
        case 1: printf ("\t\tVoce escolheu Mamao.\n");
                break;
        case 2: printf ("\t\tVoce escolheu Abacaxi.\n");
                break;
        case 3: printf ("\t\tVoce escolheu Laranja.\n");
                break;
    }
    return(0);
}
```

Algoritmos e Programação

Loops de Repetição – `do... while`

- Repete enquanto a condição for verdadeira
- A condição é testada
 - Depois da execução do bloco (Pós-teste)
- O bloco é executado pelo menos uma vez
- `do ... while` com apenas um comando no corpo não precisa de chaves
 - `do scanf ("%d", &x) while (x != 10);`

Algoritmos e Programação

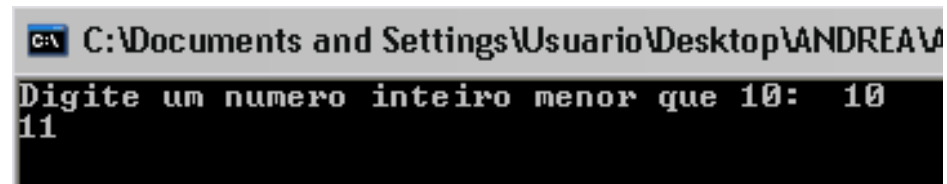
Loops de Repetição – do... while

```
#include <conio.h>
#include <stdio.h>

int main () {
    int num;
    printf("Digite um numero inteiro menor que 10:\t");
    scanf("%d" ,&num);
    do { num=num+1;
        printf("%i ",num);}
    while (num<10);
    getch();
    return 0; }
```



```
C:\Documents and Settings\Usuario\Desktop\ANDREA\
Digite um numero inteiro menor que 10: 2
3 4 5 6 7 8 9 10 _
```



```
C:\Documents and Settings\Usuario\Desktop\ANDREA\
Digite um numero inteiro menor que 10: 10
11
```



```
C:\Documents and Settings\Usuario\Desktop\ANDREA\
Digite um numero inteiro menor que 10: 12
13
```

Algoritmos e Programação

Loops de Repetição – **for**

Sintaxe

```
for (inicialização; condição; incremento) {  
    instrução;  
}
```

- Podemos omitir qualquer um dos elementos do **for**:
(inicialização; condição; incremento).

```
// int numero;  
for (int numero=1; numero<=100; numero++) {  
    printf ("%d ", numero);  
}
```

Algoritmos e Programação


Loops de Repetição – **for**

- **for** (inicialização; condição; iteração) {...};
- **Inicialização**
 - Executada uma vez, no início do for
- **Condição**
 - Controle de laço avaliada antes de cada execução do corpo
- **Iteração**
 - Executada após cada execução do corpo
 - Incrementar contador
- O contador (local) pode ser criado dentro do **for** → `int i = 0;`
- Inicia-se os contadores com zero porque as estruturas da linguagem são “base zero”
- Não altere o contador também dentro do bloco
 - `(printf (“%d, ”, i++);)`
- **for** com apenas um comando no corpo não precisa de chaves
 - `for (int i = 0; i < 9; i++) printf (“%d, ”, i);`

Algoritmos e Programação

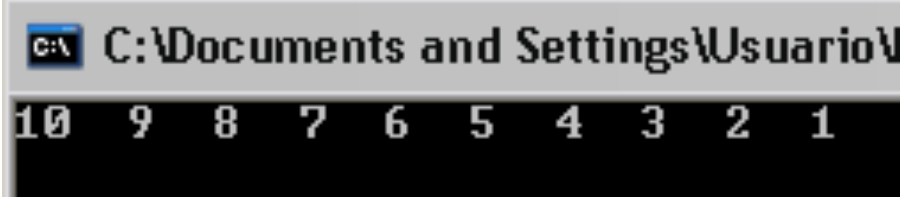
Loops de Repetição – for

```
#include <stdio.h>
int main () {
int a;
for (a=0; a<=10; a++)
    {printf ("%d ", a);}
getchar();
return (0);}
```



```
C:\Documents and Settings\Usuario\Desktop
0 1 2 3 4 5 6 7 8 9 10
```

```
#include <stdio.h>
int main () {
int a;
for (a=10; a>=1; a--)
    {printf ("%d ", a);}
getchar();
return (0);}
```

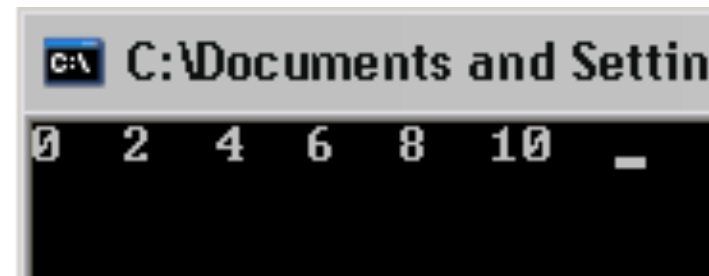


```
C:\Documents and Settings\Usuario\Desktop
10 9 8 7 6 5 4 3 2 1
```

Algoritmos e Programação

Loops de Repetição – for

```
#include <stdio.h>
int main () {
int a;
for (a=0; a<=10; a=a+2)
    {printf ("%d ", a);}
getchar();
return (0);}
```



```
C:\Documents and Settings
0 2 4 6 8 10 _
```


Algoritmos e Programação

Loops de Repetição – comando **break**

- Normalmente é utilizado em laços em que uma condição especial pode provocar uma terminação imediata.
- Faz com que a execução do programa continue na primeira linha seguinte ao loop ou bloco que está sendo interrompido.
- Utilizados para interromper os comandos: **“switch”, “for”, “while” e “do while”**.

Exemplo

```
for(;;) {  
    printf("%d", count);  
    count++;  
    if(count==10) break; }
```

Algoritmos e Programação

Loops de Repetição – comando `break`

```
#include <stdio.h>
int main ()
{
    int Count;
    char ch;
    printf(" Digite uma letra - <x para sair> ");
    for (Count=1;;Count++)
    {
        scanf("%c", &ch);
        if (ch == 'x') break;
        printf("\nLetra: %c \n",ch);
        scanf("%c", &ch);
    }
    return(0);
}
```

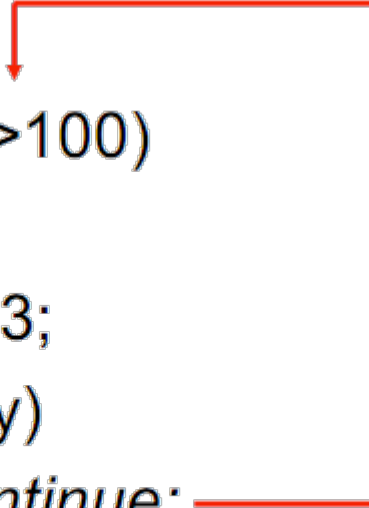
Gera um “Loop infinito”

Algoritmos e Programação

Loops de Repetição – comando `continue`

- Funciona apenas dentro de um loop;
- Quando o comando `continue` é encontrado, o loop pula para a próxima iteração, sem o abandono do loop;

```
while (x>100)
{
    x-=b*3;
    if (x<y)
        continue;
    x-=y*3;
}
```



Algoritmos e Programação

```
#include <stdio.h>
int main()
{
    int opcao;
    while (opcao != 4)
    {
        printf("\n\n Escolha uma opcao entre 1 e 4: ");
        scanf("%d", &opcao);
        if ((opcao > 4)|| (opcao <1)) continue;
        switch (opcao)
        {
            case 1: printf("\n --> Primeira opcao..");
                    break;
            case 2: printf("\n --> Segunda opcao..");
                    break;
            case 3: printf("\n --> Terceira opcao..");
                    break;
            case 4: printf("\n --> Abandonando..");
                    break;
        }
    }
    return(0);
}
```

Opção invalida:
volta ao inicio do
loop

Algoritmos e Programação

Loops de Repetição – comando `goto`

- Realiza um salto incondicional para um local determinado por um rótulo.
- Tende a tornar o código confuso. Uso não recomendado.

Sintaxe

nome_do_rótulo:

....

goto nome_do_rótulo;

Exemplo:

início_do_loop:

if (condição)

{

instrução;

incremento;

goto início_do_loop;

}