

Linguagem C – Controle de Fluxo

Objetivos

Estudar os diversos tipos de controles de fluxo disponíveis na linguagem C.

Controle de Fluxo

Os comandos de controle de fluxo podem ser divididos em três grupos: Instruções condicionais, estrutura de repetição e os desvios incondicionais que serão apresentados a seguir.

Instruções Condicionais

As instruções condicionais existentes no **padrão ANSI** são: **if**, **switch** e o operador ternário (?).

Instrução If

Esta instrução obedece à sintaxe:

If (condição) instrução para condição **verdadeira**;

Else instrução para condição **falsa**;

Pode-se utilizar uma ou mais instruções verdadeiras e falsas no if. Caso seja mais de uma deve-se colocar entre chaves. Se a condição é verdadeira serão executadas apenas as instruções localizadas após a instrução if. Caso contrario somente as instruções após o else serão executadas. O else é opcional conforme mostra o exemplo abaixo que efetua a soma de dois números e avisa ao usuário quando o resultado obtido for um número par.

1º Programa

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

```
Void main ()
```

```
{
```

```
int num1, num2, soma;
```

```
printf ("Digite o primeiro valor inteiro a ser somado:");
```

```
scanf ("%d", &num1);
```

```
printf ("Digite o segundo valor inteiro a ser somado:");
```

```
scanf ("%d", &num2);
```

```
soma=num1+num2;
```

```
printf ("Soma: %d", soma);
```

```
if ((soma%2)==0)
```

```
printf (" - numero par\n");
```

```
}
```

No programa acima é usada a instrução scanf que será ensinada num próximo artigo. O importante agora é saber que ela é a responsável pela entrada de dados pelo console. Abaixo pode ser visto a entrada do usuário e saída do programa.

- ✓ Digite o primeiro valor inteiro a ser somado: 2
- ✓ Digite o segundo valor inteiro a ser somado: 4
- ✓ Soma: 6 – numero par

Esse mesmo programa pode ser reescrito para que avise também ao usuário quando o resultado é um numero impar.

2º Programa

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

```
void main ()
```

```
{
```

```
int num1, num2, soma;
```

```
printf ("Digite o primeiro valor inteiro a ser somado:");
```

```
scanf ("%d", &num1);
```

```
printf ("Digite o segundo valor inteiro a ser somado:");
```

```
scanf ("%d", &num2);
```

```
soma=num1+num2;
```

```
printf ("Soma: %d", soma);
```

```
if ((soma%2)==0)
```

```
printf (" - numero par\n");
```

```
else
```

```
printf (" - numero impar\n");
```

```
}
```

Isto é feito acrescentando o else a instrução if. Desta forma, o programa verifica se o resto da soma dividido por dois é igual a 0, caso seja será impresso na tela que o valor é par. Caso contrario será impresso que a soma se trata de um numero impar.

A linguagem C padrão também permite o uso de **ifs** alinhados, obedecendo à forma:

```
If (condição)
```

```
Instrução;
```

```
Else
```

```
If (condição)
```

```
Instrução;
```

```
Else
```

```
If (condição)
```

```
Instrução;
```

Para entender melhor veja o próximo exemplo:

3º Programa

```
#include<stdio.h>

void main ()

{

float nota;

printf ("Digite o valor da nota (numero):");

scanf ("%f", &nota);

if ((nota>=8)&&(nota<=9))

printf ("Nota correspondente a A!!!\n");

else

if ((nota>=6)&&(nota<=7))

printf ("Nota correspondente a B!!!\n");

else

if ((nota>=4)&&(nota<=5))

printf ("Nota correspondente a C!!!\n");

else

if ((nota>=2)&&(nota<=3))

printf ("Nota correspondente a D!!!\n");

else

if ((nota>=0)&&(nota<=1))

printf ("Nota correspondente a E!!!\n");

else

printf ("Nota invalida!!!\n");

}
```

O operador ternário (?)

O operador? é muito utilizado no lugar da instrução if. Este operador requer três operandos e pode ser escrito da seguinte forma:

Exp1? Exp2: Exp3

Neste exemplo o usuário entra com o valor de x, se o valor de x for menor ou igual a 0 então o valor de x é incrementado de 1 e atribuído a y. De outra forma o y terá o valor de x decrementado de 1.

4º Programa

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

```
void main ()
```

```
{
```

```
int x, y;
```

```
printf ("digite um numero inteiro:\n");
```

```
scanf ("%d", &x);
```

```
y=x<=0 ? x+1: x-1;
```

```
printf ("y:%d\n", y);
```

```
}
```

Switch

Sua sintaxe é:

switch (expressão)

```
{  
  
  case constante1:  
  
    seqüência de comandos  
  
  break;  
  
  case constante2:  
  
    seqüência de comandos  
  
  break;  
  
  case constante3:  
  
    seqüência de comandos  
  
  break;  
  
  ...  
  
  default:  
  
    seqüência de comandos  
  
}
```

Esta instrução compara a expressão com todas as constantes caso seja verdadeira ela executa as seqüências de comandos daquela constante. Caso todas as alternativas sejam falsas o comando default é executado. A instrução break demonstrada acima é opcional, é ela que para a execução do **switch case**. O Padrão **ANSI** permite usar **257** comandos case. O comando **switch** compara apenas igualdades, enquanto que o **if** comparar qualquer expressão lógica ou relacional. Como exemplo será criado um menu para que o usuário escolha entre fazer a soma de dois números ou calcular a media.

5º Programa

```
#include<stdio.h>

void main ()

{

int entrada;

printf ("Digite 1 para calcular a soma de dois numeros\n");

printf ("Digite 2 para calcular a media\n");

scanf ("%d", &entrada);

switch (entrada) {

case 1:

printf ("Vc escolheu a opcao de somar dois numeros");

case 2:

printf ("Vc escolheu a opcao de calcular a media");

default:

printf ("Nenhuma das opcoes foi selecionada");

}

}
```

Estrutura de repetição

As estruturas de repetição são utilizadas para que um conjunto de instruções seja executado até que ocorra uma certa condição. O laço for diferencia dos restantes (while e do-while) por ter condições pré-definidas, ou seja, o numero de vezes a ser executada já é conhecido.

Laço for

Sintaxe do laço:

for (inicialização; condição; incremento) comando;

Na inicialização é atribuído um valor para variável que controla o laço, a condição determina quando o laço deve ser encerrado e por ultimo temos o quanto a variável controladora é incrementada. Exemplo simples:

6º Programa

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

```
void main ()
```

```
{
```

```
int n;
```

```
for (n=1; n<=10; n++)
```

```
printf ("n=%d\n", n);
```

```
}
```

No programa acima o for iniciamos a variável n com o valor, a instrução que escreve o valor de n será executada enquanto n for menor ou igual a 10, sendo que n é incrementada de 1 em iteração. Sendo assim a saída do programa será:

n=1

n=2

n=3

n=4

n=5

n=6

n=7

n=8

n=9

n=10

Laço While

Sua forma geral é:

while (condição) **Instrução**;

Este laço executa a instrução até que a condição se torne falsa (qualquer valor diferente de zero).

7º Programa

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

```
void main ()
```

```
{
```

```
int n1, n2, soma, resposta;
```

```
resposta=1;
```

```
while (resposta==1)
```

```
{
```

```
printf ("Digite valor de n1(inteiro):\n");
```

```
scanf ("%d", &n1);
```

```
printf ("Digite valor de n2(inteiro):\n");
```

```
scanf ("%d", &n2);
```

```
soma=n1+n2;
```

```
printf ("Soma:%d\n", soma);
```

```
printf ("Deseja continuar? (1 - sim / 2 - nao)\n");
```

```
scanf ("%d", &resposta);
```

```
while ((resposta!=1) && (resposta!=2))
```

```
{
```

```
printf ("Digite 1 para sim ou 2 para nao!!\n");
```

```
scanf ("%d", &resposta);
```

```
}/*fim_while*/
```

```
}/*fim_while*/
```

```
}
```

Nesse programa foram executados dois laços while, o primeiro executa a soma de dois números até que a resposta seja igual a 1. E o segundo que pede um valor válido, ou seja, 1 para continuar calculando a soma de outros números ou 2 para terminar o programa.

Laço do-while

A grande diferença entre o laço do-while e os vistos anteriormente é que ele analisa a condição no final do laço. Dessa forma todas as instruções localizadas dentro do do-while será executada pelo menos uma vez. O laço termina quando a condição se torna falsa. Sintaxe:

```
Do {  
  
Instrução;  
  
} while (condição)
```

O programa anterior também pode ser escrito da seguinte forma:

8º Programa

```
#include<stdio.h>  
  
void main ()  
  
{  
  
int n1, n2, soma, resposta;  
  
do {  
  
printf ("Digite valor de n1(inteiro):\n");  
  
scanf ("%d", &n1);  
  
printf ("Digite valor de n2(inteiro):\n");  
  
scanf ("%d", &n2);  
  
soma=n1+n2;  
  
printf ("Soma:%d\n", soma);  
  
printf ("Deseja continuar? (1 - sim / 2 - nao)\n");  
  
scanf ("%d", &resposta);
```

```

if ((resposta!=1)&& (resposta!=1))

printf ("Programa finalizado, este numero nao existe no menu \n");

} while (resposta==1);

}

```

A única diferença é que este programa permite digitar outro número além de 1 e 2, porém será exibida a mensagem “Programa finalizado, este numero não existe no menu”.

Desvio incondicional

A linguagem C tem como desvio incondicional o comando **goto**.

Comando goto

Esta instrução ou comando é muito pouco utilizado, pois tornam os códigos difíceis de compreender. Por esse motivo não entraremos em muitos detalhes. Veja o exemplo abaixo, ele faz o mesmo do que o exemplo que utilizando o **laço for** (mostrado anteriormente). Repare que o programa que utiliza o **for** é muito mais fácil de entender.

9º Programa

```

#include<stdio.h>

void main ()

{

int n;

n=1;

loop1: /*rótulo*/

printf ("n=%d\n", n);

n++;

if (n<=10) goto loop1; /*caso n seja menor ou igual a 10 volta para o rótulo
acima, executando assim as instruções outra vez
até que a condição do if seja falsa*/

}

```

Conclusão

Conclui-se que o controle de fluxo é essencial nos programas. Portanto não poderia faltar em nosso tutorial de linguagem C. Neste artigo nos aprendemos usá-los através de exemplos fáceis de entender.