{introcomp}

Working 04 : Conceitos Básicos III

Objetivos:

- Dominar a construção de estruturas de repetição em C;
- Aperfeiçoar-se na resolução de problemas;

Prazo de Envio: sábado, 16/09, 04:00.

1 INTRODUÇÃO

Finalizando os conceitos básicos, neste Working apresentaremos outras três estruturas de repetição: for, while e do-while. Além destes, vamos compreender algumas práticas de programação em processamento de textos e jogos. Por fim, vamos praticar e dominar todos os conceitos básicos que permeiam o alicerce da programação de computadores!

2 A ESTRUTURA DE REPETIÇÃO FOR

O comando de repetição condensado permite agrupar, em um único comando, a inicialização de uma variável, o incremento desta variável e o teste de parada. Seu uso mais comum em situações nas quais o número de repetições da sequência de comandos conhecido antes mesmo do início do for. A estrutura do comando é exibida a seguir:

```
for (<inicializacao>; <expressao logica>; <incremento>){
<Sequencia de comandos>
}
```

Inicialização é um comando de atribuição usado para colocar um valor na variável de controle utilizada na expressão lógica. Assim como os demais comandos de repetição, a expressão lógica representa a condição de parada das iterações. Incremento define como a variável de controle será modificada a cada iteração. Um exemplo para entendermos:

```
#include <stdio.h>
int main () //este programa imprime numeros de 1 a 'n'.
{
    int n;
    int i;


    scanf ("%d",&n); //Digitar um numero inteiro maior que zero

    for (i=1 ; i<=n ; i++) //observe que e utilizado ';'!!
        printf ("%d\n",i);

    return 0;
}
```

Observe que a variável contadora foi inicializada com o valor 1 e é incrimentada com o valor 1 (i++). Além disso, haverá a parada quando i for maior que n.

3 A ESTRUTURA DE REPETIÇÃO WHILE

O comando de repetição while é utilizado para repetições de trechos de códigos em que há a necessidade de se fazer um teste antes da execução de determinado trecho de código. Por isso, é conhecido como comando de repetição com pré-condição. Vejamos a sintaxe do comando:

```
while (<expressao logica>)
{
  <sequencia de comandos>
}
```

A expressão lógica presente entre parênteses definirá a condição de parada do comando, ou seja, enquanto a expressão lógica for verdadeira, a sequência de comandos que estiver entre as chaves continuará a ser repetida.
Vejamos um exemplo de utilização do comando:

```
#include <stdio.h>
int main()
{
  int num = 2;
  while (num%2 == 0)
  {
      scanf ("%d", &num);
      printf ("%d\n", num);
  }
  return 0;
}
```

O trecho de código acima permite que o usuário imprima números enquanto esses números forem pares, e os imprime na tela.

4 A ESTRUTURA DE REPETIÇÃO DO-WHILE

Em contraste com o comando de repetição com pré-condição, o comando de repetição com pós-condição, do while, só efetua o teste da expressão lógica (condição de parada) após a primeira execução da sequência de comandos. Logo, a sequência de comandos é executada pelo menos uma vez, independente da expressão lógica. A própria sintaxe do comando sugere essa diferença:

```
do{
<sequencia de comandos>
} while (<express o l gica>);
```

Um exemplo para entender sua utilidade:

```
#include <stdio.h>
int main ()
{
    int candidato;
    int confirma;

    do{
        printf ("Digite o numero do seu candidato: ");
        scanf ("%d",&candidato);
        switch (candidato)
```

```
11          {
                case 13:
13                printf ("Voce votou na Iriny Lopes!\n");
                break;
15              case 23:
                  printf ("Voce votou no Rezende!\n");
17              break;
                case 43:
19                printf ("Voce votou no Luiz Paulo!\n");
                break;
21              default:
                printf ("Voce votou nulo!\n");
23          }
            printf ("Digite 1 (confirmar) ou 0 (votar novamente): ");
25          scanf ("%d",&confirma);
        }while(!confirma);
27
        return 0;
29  }
```

O programa acima permite ao usuário escolher um candidato a partir da digitação de seu número, e finaliza a repetição quando o usuário decide confirmar seu voto, ou seja, digitar 1 após a escolha do candidato. Caso o usuário deseje modificar seu voto, basta escolher 0 para votar novamente, e desta maneira a repetição continua por pelo menos mais um ciclo.

5 DESVIO INCONDICIONAL BREAK

```
1  while (<condicao>)
   {
3    for (inicializacao; condicao; incremento)
     {
5        while (<condicao>)
         {
7          if (<condicao>)
             break; // Ocorre a interrupcao da estrtura de repeticao, seguindo com o decorrer
       do codigo.
9          ...
           ...
11         ...
         }
13       ...
         ...
15       ...

17   }
   }
```

O comando break desvia o fluxo do programa de forma que a execução do mesmo continua no bloco de código do loop exterior. Caso não haja um loop exterior, o programa é desviado para o bloco de código exterior.

6 DESVIO INCONDICIONAL CONTINUE

```
while (<condicao>)
{
  for (inicializacao; condicao; incremento)
  {
      while (<condicao>)
      {
        if (<condicao>)
          continue;
        ...
        ...
        ...
      }
      ...
      ...
  }
}
```

O comando continue desvia o fluxo do programa de forma que a execução do mesmo continua ao final do bloco de código do loop atual.

Praticando

Agora vamos praticar! Para todos os praticando em que é pedido que se escreva um programa, escreva o código do seu programa e nos envie o .c (código fonte) correspondente.

1. Considere o código abaixo:

```
#include <stdio.h>
int main()
{
    int i,j,cont;
    for(i=0;i<10;i++)
    {
      if(i==10){
          break;
        }

        for(j=0;j<20;j++)
        {
          if(i>=9)
          {
            break;
          }
        }
        cont = cont + 2;
    }
    return 0;
}

```

   (a) Para que linha do código acima, o comando break da linha 15 desviará o fluxo de execução do programa?

   (b) O comando break da linha 8 é inútil. Explique por quê.

   (c) Descreva brevemente o funcionamento do comando break.

2. Considere os códigos a seguir:

(a)

```
        for(i=0;i<20;i++)
        {
            //Codigos...
        }

```

(b)

```
        i=0;
        while(i<100)
        {
            //Codigos..
            i = i + 2;
        }

```

(c)

```
for(i=0,j=2;i<8 || j<16;i++,j+=2)
{
//Codigos
}
```

Transcreva os códigos de cada letra por alguma outra estrutura de repetição dentre as possíveis (do-while, while e for), que não esteja já sendo utilizada pela letra.

3. Faça um programa em C que calcula o produto dos números digitados pelo usuário. O programa em C deve permitir que o usuário digite uma quantidade não determinada de pares de números. E que encerre quando o usuário digita 0. Use as estruturas que achar apropriado.

Exemplo de Entrada	Exemplo de Saida
2 5 0	10
5 5 2 5 0	25 10

4. Faça um programa que simule um jogo de loteria: deverá ser pré estabelecido um número vencedor pelo programador, e o usuário deverá tentar adivinhá-lo. O usuário irá digitar diversos números até acertar o número definido. O programa deverá informar a quantidades de números digitados até acertar o número pré-definido.

Exemplo de Entrada	Exemplo de Saida
77 4 5 77	3
13 1 2 3 4 13	5

5. Exiba a tabuada de multiplicação de ‘n’.
Ex:

Exemplo de Entrada	Exemplo de Saida
1	1*1 = 1 1*2 = 2 1*3 = 3 1*4 = 4 1*5 = 5 1*6 = 6 1*7 = 7 1*8 = 8 1*9 = 9 1*10 = 10

Desafios

1. Faça um programa em C que, dado dois números inteiros lidos pelo teclado imprima o MDC (Máximo Divisor Comum) desses dois números.

Exemplo de Entrada	Exemplo de Saida
25 10	5
54 182	2

NO PRÓXIMO ENCONTRO...
No próximo encontro vamos iniciar os estudos em modularização na linguagem C!