Algoritmos I

LAÇO DE REPETIÇÃO

Laço de Repetição

- Imagine que fosse necessário solicitar 5 valores ao usuário, somá-los todos e exibir o resultado da soma.
- Se fosse utilizado algoritmo sequencial seria necessário criar 5 variáveis, fazer a leitura das mesmas e somá-las.

```
1   #include <stdio.h>
2
3   int main() {
4       int num1, num2, num3, num4, num5;
5
6       scanf("%d%d%d%d%d",&num1,&num2,&num3,&num4,&num5);
7
8       int soma = num1 + num2 + num3 + num4 + num5;
9       printf("\nSoma: %d",soma);
10  }
```

Laço de Repetição

- Observando o código com atenção, nota-se que existem instruções que se repetem: a leitura e a soma dos valores.
- Sempre que existir repetição de código é utilizado Laço de Repetição.
- Portanto, Laço de Repetição é uma estrutura que permite repetir um determinado bloco de instruções até que uma (ou mais) condição(ões) seja(m) satisfeita(s).
- Existem três tipos de laço de repetição:
  - Laço de repetição com teste lógico no início.
  - Laço de repetição com teste lógico no final.
  - Laço de repetição com variável de controle.

Laço de Repetição com teste lógico no início

- Neste laço a condição de término é definida desde o início.
- Ele testa a condição (ou condições) e caso seja(m) verdadeira(a) executa o bloco de instruções que seguem após a condição, entre as chaves. Caso seja(m) falsa(s) executam a primeira instrução fora do laço (depois do fechamento das aspas).
- Sintaxe:

C

```
while (condição/ões) {
        ... Bloco de Instruções ...
}
```

Laço de Repetição com teste lógico no início

- Exemplo. Elabore um algoritmo que exiba os valores de 1 a 10 na tela.

```
1   #include <locale.h>
2   #include <stdio.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int contador = 1;
8
9       while (contador <= 10) {
10          printf("%d\n",contador);
11          contador = contador + 1;
12      }
13
14      return 0;
15  }
```

Enquanto esta condição for verdadeira o código entre as chaves será executado.

Laço de Repetição com teste lógico no início

- Dentre os problemas que podem ocorrer, um deles é o conhecido como looping infinito.

```
1   #include <locale.h>
2   #include <stdio.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int contador = 1;
8
9       while (contador <= 10) {
10          printf("%d\n",contador);
11      }
12
13      return 0;
14  }
```

Laço de Repetição com teste lógico no início

- Neste caso a condição sempre será verdadeira, fazendo com que o laço execute sem parada. A variável contador vale somente 1.
- Isso ocorreu pois não existe nenhuma instrução que faça com que a variável contador seja 10 em algum momento.
- Seria necessário uma instrução chamada contador.
- O contador nada mais é que uma variável a qual somamos um outro valor numérico e atribuímos para a própria variável (para modificar a própria variável e na próxima iteração trabalhar com o novo valor).

Laço de Repetição com teste lógico no início

- Ao se trabalhar com um contador é necessário atribuir um valor inicial a mesma, antes de executar a condição do laço de repetição.
- Isso ocorre pois, ao declararmos uma variável a mesma possui o que é chamado lixo de memória, ou seja, possui um valor qualquer nela.
- Então, no caso era necessário imprimir o valor apartir de 1, então, foi atribuido 1 a variável. Se o início fosse apartir de 30, a variável seria inicializada de 30 (contador = 30).
- Se fosse necessário gerar os números de 2 a 2 (1, 3, 5, 7, 9, 11, 13, 15, 17, 19) o contador seria modificado para **contador = contador + 2;**

Laço de Repetição com teste lógico no início

- Existem as reduções que podem ser utilizadas. Seguem abaixo a lista.

Incrementos	Explicação

variavel = variavel + 1;
variavel += 1;
variavel++; (Redução)
++variavel; (Redução)
	Todas as 3 fazem a mesma função: incrementam 1 a variável.

variavel = variavel + 5; variavel += 5; (Redução)	As duas incrementam 5 a variável.
variavel = variavel % 7; variavel %= 7; (Redução)	É efetuado o resto da divisão inteira com o valor contido em variável por 7, e atribuído a variávell
variavel = variavel + numero; variavel += numero; (Redução)	As duas incrementam o valor que tiver na variável número.

Laço de Repetição com teste lógico no início

- As reduções podem ser efetuadas com qualquer operador aritmético.
- Existem dois contadores que apesar de parecerem iguais possuem diferenças. Segue a explicação:
  - cont++;
  - ++cont;
- Ambos fazem o mesmo processo, que é incrementar a variável cont de 1. Segue abaixo a explicação da diferença:
  - cont++;
    - Utiliza a variável e incrementa o valor depois.
    - Por exemplo, o código printf(“%d”, cont++);
      - Imagine que foi atribuído 6 à variável cont. Neste caso imprimiria 6 e depois incrementaria mais um, e a variável cont conteria 7.

Laço de Repetição com teste lógico no início

- ++cont;
  - Incrementa o valor e depois utiliza a variável.
  - Por exemplo, o código printf(“%d”,++cont);
    - Imagine que foi atribuído 6 à variável cont. Neste caso incrementaria mais um, e a variável cont conteria 7. Depois imprimiria 7.
- Na tabela de incrementos houve um código diferenciado -> variavel = variavel + numero;
  - Este caso seria chamado de acumulador ou somador, pois a variavel soma os valores.
- O somador ou acumulador pode ser utilizado com qualquer operador aritmético.
- Abaixo, segue um exemplo da utilização do somador.

Laço de Repetição com teste lógico no início

- Faça um programa que solicite o valor de 10 salários e informe, ao final, a soma de todos.

```
1   #include <locale.h>
2   #include <stdio.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int cont = 0;
8       float salario, somaSalarios = 0;
9
10      while (cont < 10) {
11          printf("\nEntre com o salário: ");
12          scanf("%f",&salario);
13
14          somaSalarios = somaSalarios + salario;
15          cont++;
16      }
17
18      printf("\n\nA soma dos salários é %.2f",somaSalarios);
19
20      return 0;
21  }
```

A variável somaSalarios está acumulando os valores lidos na variável salario. Por este motivo foi inicializada por 0.

Laço de Repetição com teste lógico no início

- Faça um programa que gere a tabuada de 1 a 10.

```
1  #include <locale.h>
2  #include <stdio.h>
3
4  int main() {
5      setlocale(LC_ALL,"Portuguese");
6
7      int operando1 = 1, operando2, calculo;
8
9      while (operando1 <= 10) {
10         operando2 = 1;
11         while (operando2 <= 10) {
12             calculo = operando1 * operando2;
13             printf("\n%d * %d = %d",operando1,operando2,calculo);
14             operando2++;
15         }
16         operando1++;
17         printf("\n\n");
18     }
19
20     return 0;
21 }
```

Quando se tem um laço dentro do outro, ele executa uma vez o laço mais externo (passa a valer 1) e depois todo o laço externo (varia de 1 a 10). Isto será feito até que o laço externo passe a valer 11. A toda vez que o laço externo rode uma vez, o interno roda 10 vezes (neste caso).

Laço de Repetição com teste lógico no final

- A diferença entre o laço de repetição com teste lógico no final e o while(){ ...} é que ele executa ao menos uma vez, pois o teste lógico é executado ao final.
- Todas as regras do while valem para o do ... while.
- Uma das situações mais utilizadas é para validação de informação. Por exemplo, garantir que um valor digitado seja maior que 0.
- Sintaxe:

C

```
do {
        ... Bloco de Instruções ...
} while (condição/ões);
```

Laço de Repetição com teste lógico no final

- Faça um algoritmo que leia um valor e garanta que o mesmo seja positivo.

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

int main() {
    setlocale(LC_ALL,"Portuguese");

    int numero;

    do {
        printf("Qual é o número? ");
        scanf("%d",&numero);
        if(numero <= 0){
            printf("\nValor inválido! Deve ser digitado um valor maior que 0.\n\n");
        }
    }while (numero <= 0);

    printf("\n\n O valor digitado foi %d",numero);

    return 0;
}
```

Laço de Repetição com teste lógico no final

- Elabore um algoritmo que exiba os valores de 1 a 20 na tela.

```
1   #include <stdio.h>
2
3   int main() {
4       int cont = 1;
5
6       do {
7           printf("%d ",cont);
8           cont++;
9       } while (cont <= 20);
10
11      return 0;
12  }
```

Laço de Repetição com teste lógico no final

- Faça um programa que gere a tabuada de 1 a 10.

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

int main() {
    setlocale(LC_ALL,"Portuguese");

    int operando1 = 1, operando2, calculo;

    do {
        operando2 = 1;
        do{
            calculo = operando1 * operando2;
            printf("\n%d * %d = %d",operando1,operando2,calculo);
            operando2++;
        }while (operando2 <= 10);
        operando1++;
        printf("\n\n");
    }while (operando1 <= 10);

    return 0;
}
```

Laço de Repetição com teste lógico no final

- Sempre existe confusão entre o while e o do ... while. O principal é o que o while realiza o teste lógico no início, ou seja, pode não executar. Já o do ... while sempre executará uma vez, pois o teste será somente ao final do laço.
- O do ... while é utilizado quando se precisa executar o laço uma vez ao menos, ou quando não é preciso testar a primeira vez (em muitos casos que se pode usar o while() pode-se utilizar também do do ... while.
- O laço será executado enquanto a condição for verdadeira. Apartir do momento que for falsa ele executa a primeira instrução fora do laço.

Laço de Repetição com variável de controle

- O laço de repetição for funciona como os outros dois. Enquanto a condição é verdadeira o laço é executado. Só para a execução quando a condição se torna falsa.
- Sintaxe:

C

```
for (inicialização da(s) variável(is); condição(ões); incremento/decremento) {
                    ... Bloco de Instruções ...
}
```

Laço de Repetição com variável de controle

- O laço de repetição for é formado pelos itens abaixo:
  - Inicialização: atribui os valores de início das variáveis.
  - Condição: condição ou condições para que o laço execute.
  - Incremento/Decremento: incremento/decremento de variáveis.
- O que separa cada um dos itens é o ponto e vírgula (;).
- O laço for pode conter os três itens ou não.

Laço de Repetição com variável de controle

- Elabore um algoritmo que exiba os valores de 1 a 20 na tela.

```
1   #include <stdio.h>
2   #include <locale.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int cont;
8
9       for (cont=1; cont <= 20; cont++) {
10          printf("\n%d",cont);
11      }
12
13      return 0;
14  }
```

Inicializou a variável cont de 1.

Toda vez que o laço repete incrementa 1 a variável cont.

Enquanto a variável cont não chegar a 21, o laço continuará repetindo.

Laço de Repetição com variável de controle

- Elabore um algoritmo que exiba os valores de 10 a 100 na tela, pulando de 5 em 5.

```
1   #include <stdio.h>
2   #include <locale.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int cont;
8
9       for (cont=10; cont <= 100; cont = cont + 5){
10          printf("\n%d",cont);
11      }
12
13      return 0;
14  }
```

Incrementa a variável cont de 5 em 5.

Laço de Repetição com variável de controle

- Faça um programa que gere a tabuada de 1 a 10.

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

int main () {
    setlocale(LC_ALL,"Portuguese");

    int operando1, operando2, calculo;

    for (operando1 = 1; operando1 <= 10; operando1++){
        for (operando2 = 1; operando2 <= 10; operando2++){
                calculo = operando1 * operando2;
                printf("\n%d * %d = %d",operando1,operando2,calculo);
        }
        printf("\n\n");
    }

    return 0;
}
```

Laço de Repetição com variável de controle

- Em relação aos itens que constituem o for, pode-se ter os 3, bem como algum deles pode ser excluído.
- Também pode-se ter mais de um item, que deve ser separado por vírgula (,).

```
1   #include <stdio.h>
2   #include <locale.h>
3
4   int main() {
5       setlocale(LC_ALL,"Portuguese");
6
7       int numero1, numero2;
8
9       for (numero1 = 1, numero2 = 0; numero2 <= 100; numero1++, numero2 += 5){
10          printf("%d\t%d\n",numero1,numero2);
11      }
12
13      return 0;
14  }
```