PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

Modulo 4: while – do …while
(limites de variáveis)

Aura - Erick
aconci@ic.uff.br, erickr@id.uff.br

Roteiro

- Até aqui apreendemos as estruturas:
- If e If...else
- Switch ( ) { } - case , break , default
- For (; ;) { ….}
- Continuando as estruturas de controle de fluxo com 2 formas de **Repetição incontável:**
  - while
  - do …while

Falaremos ainda de área de armazenamento e limites de variáveis em C

Revisão – Repetição contável - For

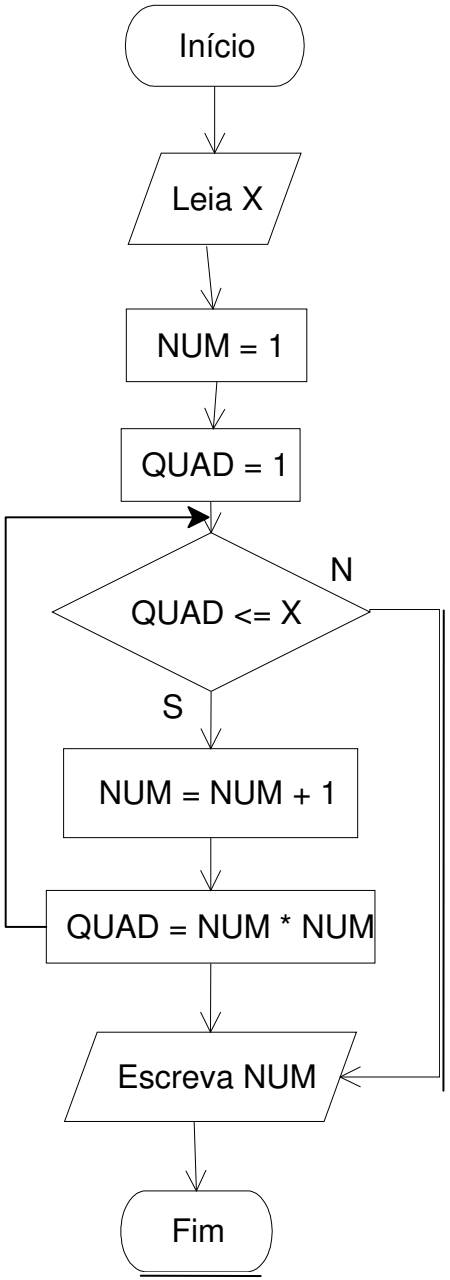
- Usada quando sabemos quantas vezes queremos repetir um comando (ou bloco de comandos)

- Não resolve todos os casos

Exemplo

- Calcular o menor inteiro positivo cujo quadrado é superior a um número -X lido (inteiro e positivo)

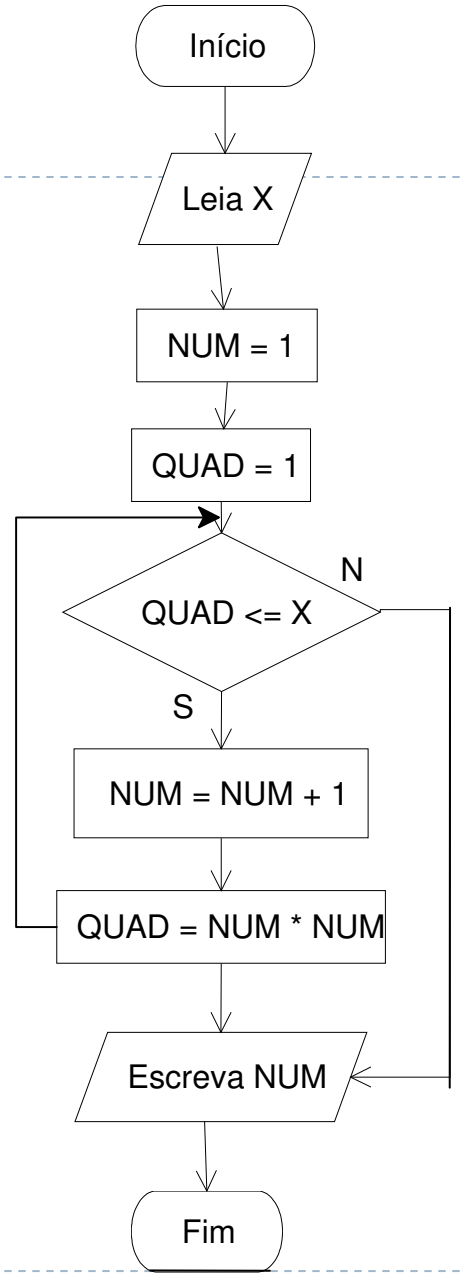
- Dá para saber quantas repetições serão necessárias?
  - Não, o critério de parada depende do teste de uma condição (o quadrado do número ser maior que X), que só será conhecido quando o programa estiver sendo usado ou executado por alguém (usuário) .



Calcular o menor inteiro positivo cujo quadrado é superior a um NUMero - X lido (inteiro e positivo)

Algoritmo

1. Leia X
2. NUM = 1
3. QUAD = 1
4. Enquanto QUAD <= X Faça
    1. NUM = NUM + 1
    2. QUAD = NUM * NUM
5. Escreva NUM



Mas como esse pseudocódigo ou fluxograma pode ser escrito em C?

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

Funcionamento

1. A expressão é avaliada;
2. Se a expressão for VERDADEIRA então o comando é executado, caso contrário a execução do comando é terminada;
3. Voltar para o passo 1.

Se a expressão for FALSA já na primeira vez, o comando não será executado NENHUMA VEZ

Exemplo em C

```
#include <stdio.h>

main()
{
      int X,NUM,QUAD;
      printf("Digite um numero inteiro");
      scanf("%d", &X);
      NUM = 1;
      QUAD = 1;
      while (QUAD <= X)
      {
         NUM++;
         QUAD = NUM * NUM;
      }
      printf("O numero eh %d", NUM);
      system("PAUSE");
}
```

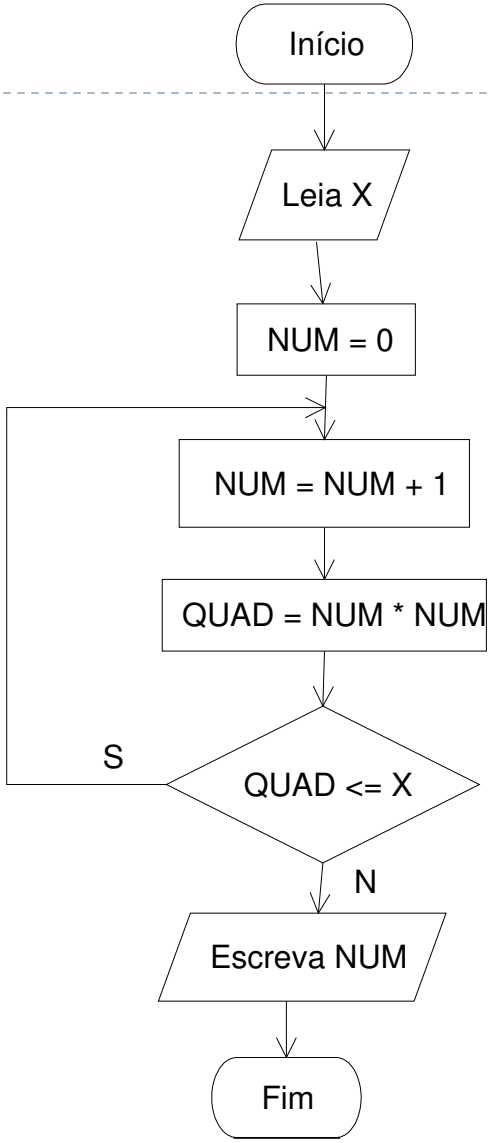
Comando while

- O comando WHILE testa a condição ANTES de executar o comando

- Mas em alguns casos, queremos testar a condição DEPOIS de executar o(s) comando(s), isso é executar pelo menos 1 vez ! ! !

Mesmo exemplo anterior

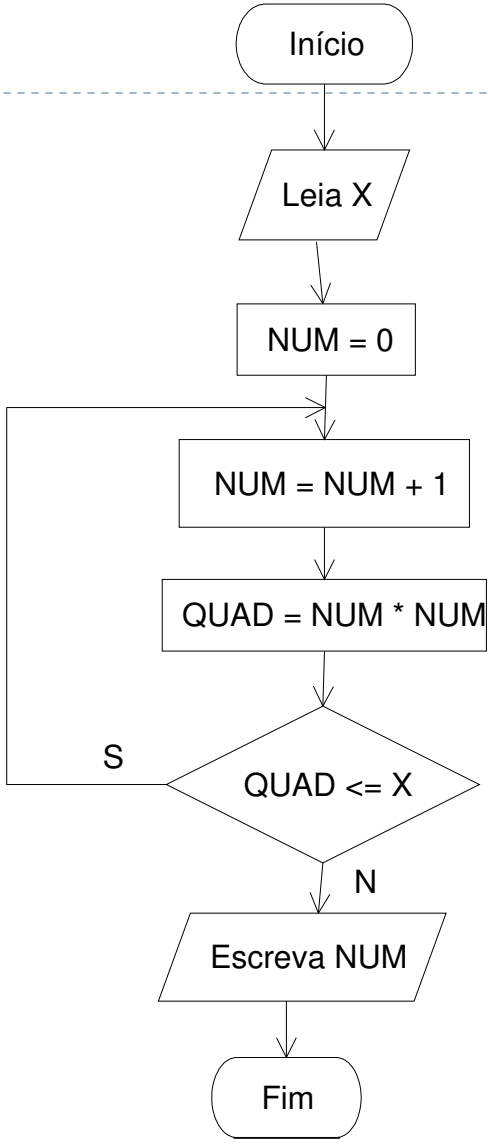
- Calcular o menor inteiro positivo cujo quadrado é superior a um X lido (inteiro e positivo)



Algoritmo

1. Leia X
2. NUM = 0
3. Faça
    1. NUM = NUM + 1
    2. QUAD = NUM * NUM

    Enquanto QUAD <= X
4. Escreva NUM



Em C isso é feito por ...

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

Funcionamento

1. Executa o comando;
2. Avalia a expressão;
3. Se a expressão for VERDADEIRA então volta para o passo 1, caso contrário interrompe o do-while

Exemplo em C

```
#include <stdio.h>

main()
{
      int X, NUM, QUAD;
      printf ("Digite um numero inteiro");
      scanf ("%d", &X);
      NUM = 0;
      do {
         NUM++;
         QUAD = NUM * NUM;
      } while (QUAD <= X);
      printf ("O numero eh %d", NUM);
      system ("PAUSE");
}
```

Exemplo

- Escreva um algoritmo que calcule a média dos números digitados pelo usuário, se eles forem pares. Termine a leitura se o usuário digitar zero (0)

- Solução usando **while**

```
#include <stdio.h>
#include <stdlib.h>
main(){
int num, soma=0, cont=0;
float media;
printf ("\nDigite um  numero inteiro. Digite zero para encerrar a execucao: ");
scanf ("%d", &num);
while (num!=0)
{
  if (num%2==0){
    soma=soma+num;
    cont++;
  }
  printf ("\nDigite um  numero inteiro. Digite zero para encerrar a
    execucao: ");
  scanf ("%d", &num);
 }
if (cont>0){
   media=(float)soma/cont;
   printf ("\nA media eh %.2f\n", media);
 }
else
    printf ("\nNenhum valor foi digitado");
system("pause");
}
```

```
#include <stdio.h>
#include <stdlib.h>
main(){
int num, soma=0, cont=0;
float media;
printf ("\nDigite um  numero inteiro. Digite zero para encerrar a execucao: ");
scanf ("%d", &num);
while (num!=0)
{
  if (num%2==0){
    soma=soma+num;
    cont++;
  }
  printf ("\nDigite um  numero inteiro. Digite zero para encerrar a
    execucao: ");
  scanf ("%d", &num);
 }
if (cont>0){
   media=(float)soma/cont;
   printf ("\nA media eh %.2f\n", media);
 }
else
    printf ("\nNenhum valor foi digitado");
system("pause");
}
```

Outra solução

- Agora com **do … while**

```
#include <stdio.h>
#include <stdlib.h>
main(){
int num, soma=0, cont=0;
float media;
printf ("\nDigite um  numero inteiro. Digite zero para encerrar a execucao: ");
scanf ("%d", &num);
if (num!=0)
{
   do {
      if (num%2==0)
         {
               cont++;
               soma=soma+num;
         }
     printf ("\nDigite um  numero inteiro. Digite zero para encerrar a execucao: ");
     scanf ("%d", &num);
   } while (num!=0);
   media=(float)soma/cont;
   printf ("\nA media eh %.2f", media);
}//if
system("pause");
}
```

Organização de computadores

- A memória de qualquer computador é uma seqüência de bytes.
- Cada *byte* consiste em 8 bits (= dígitos binários) e portanto tem $2^8 = 256$ possíveis valores:
- Que vão de:

00000000,

00000001,

00000010,

. . . ,

11111110,

11111111 (255 em representação decimal) .

a *notação binária*:

cada seqüência de bits representa o resultado da soma das potências de 2 que correspondem aos bits 1.

Por exemplo, a seqüência

- 1101 representa o número 13, pois
- $1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 =$
- $8 + 4 + 1 = 13$.

O conjunto de todas as sequências de *k* bits representa os números naturais de 0 a $2^k - 1$.

limites de variáveis

- área de armazenamento x tamanho de cada tipo de dado

- Qual os ***ranges*** dos valores que podem ser armazenado nos tipos de dados?

Na linguagem C,

- **números naturais** (0, 1, 2, 3, etc.) são conhecidos como **inteiros sem sinal**.
- Esse é um dos tipos de dados básicos da linguagem C.
- Uma variável n desse tipo é declarada assim:

unsigned int n;

- (No lugar de **unsigned int** podemos escrever, simplesmente, **unsigned.)**
- Um **unsigned int** é armazenado em ***s*** bytes consecutivos, sendo ***s*** o valor da expressão **sizeof (unsigned int).**
- Portanto, um **unsigned int** é representado por **8*s*** bits e assim pode assumir $\mathbf{2^{8s}}$ valores, a saber: 0, 1, . . . , $2^{8s}-1$.

Na linguagem C,

- Em alguns computadores, ***s*** vale **2** e portanto um **unsigned int** tem $\mathbf{2^{16}}$ valores, de 0 a **65535**.
- Em muitos computadores, ***s*** vale **4** e assim um **unsigned int** tem $\mathbf{2^{32}}$ valores, que representam os números 0 a **4294967295**.

- O número $\mathbf{2^{8s}-1}$ é o valor da constante **UINT_MAX**, definida na interface **limits.h**.

Exemplo de limites de inteiros

- **short int** e **int**: -32,767 to 32,767
- **unsigned short int** e **unsigned int**: 0 to 65,535
- **long int**: -2,147,483,647 até 2,147,483,647
- **unsigned long int**: 0 to 4,294,967,295
- **long long int**: -9,223,372,036,854,775,807 até 9,223,372,036,854,775,807
- **unsigned long long int**:
  - 0 até 18,446,744,073,709,551,615

Exercícios

1. Ler os valores de 3 notas por aluno, imprimir média de cada aluno e número de alunos. Depois do último aluno, nota 1 = -1 (condição de parada).

x x x
x x x
x x x
-1 x x

- 3 alunos devem ser considerados

Exercícios

2. Refaça o programa anterior para que ele leia, além das notas, o código do aluno. O programa pára quando o código do último aluno for 234. Veja que este aluno deve ser incluído na contagem.

Exercícios

3. Se quiséssemos usar FOR, teríamos que saber a princípio para quantos alunos a média deverá ser calculada. Refaçam o programa para que ele se comporte desta forma.

Exercícios

- Façam todos os exercícios sugeridos nos módulos, como uma forma de você estudar para nossa P1.
- Em caso de dúvidas procure logo ajuda do Erick, mandando uma e-mail e se preciso marcando horário para atendimento
- Não deixe para estuda na véspera, assim de uma hora pra outra, não se apreende direito pois o aprendizado é um processo constante e incremental ! !

Para fixar o estudado até aqui

Terceiro Trabalho : (Entrega: 12/01/2016)

Você deve usar neste pelo menos uma vez cada uma das 4 estruturas de controle de fluxo vistas até aqui (if, if else, for, switch). No inicio apresente um texto explicando que dependendo da letra teclada ser A/a, D/d, E/e, G/g, M/m, ou P/p , 6 opções de funções serão executadas pelo programa. Dê um aviso de "erro" caso nenhuma destas tenha sido escolhida. A/a corresponde a ser calculado e mostrado na tela a área de um círculo e será pedido para o usuário "teclar" o raio do mesmo. P/p corresponde a ser mostrado o perímetro de um circulo e será pedido para o usuário "teclar" o raio do mesmo. M/m corresponde a ser mostrado o maior entre 4 números "teclados" pelo usuário. G/g mostra ao usuário o número google. E/e calcula o quadrado da distância Euclidiana entre 2 pontos do plano fornecidos. D/d calcula a distância ao quadrado entre 2 pontos do plano fornecidos pelo usuário na forma da distância descrita no modulo 3.

Mandem ele (código e executável renomeando o .exe para .trab2) por e-mail para o "erickr@id.uff.br". No subject da e-mail - incluir PROG V - TRAB 3.
Partes de trabalhos iguais terão nota Zero !

Bibliografia

- http://www.ime.usp.br/~pf/algoritmos/aulas/int.html