Linguagem de Programação II

Prof. Mario Bessa

Aula 6

http://mariobessa.info

Comandos Repetitivos (laços)

- Consideremos um programa que calcula e imprime a média das notas N1, N2 e N3 de um aluno:

```
passo 1: scanf("%f%f%f",&N1,&N2,&N3);
passo 2: MEDIA = (N1 + N2 + N3) / 3;
passo 3: printf("%f", MEDIA);
```

Comandos Repetitivos (laços)

- Se quisermos calcular a média de 2 alunos:

passo 1: scanf(“%f%f%f”,&N1,&N2,&N3);
passo 2: MEDIA = (N1 + N2 + N3) / 3;
passo 3: printf(“%f”, MEDIA);
passo 4: scanf(“%f%f%f”,&N1,&N2,&N3);
passo 5: MEDIA = (N1 + N2 + N3) / 3;
passo 6: printf(“%f”, MEDIA);

Comandos Repetitivos (laços)

passo 1: scanf(“%f%f%f”,&N1,&N2,&N3);

passo 2: MEDIA = (N1 + N2 + N3) / 3;

passo 3: printf(“%f”, MEDIA);

passo 4: scanf(“%f%f%f”,&N1,&N2,&N3);

passo 5: MEDIA = (N1 + N2 + N3) / 3;

passo 6: printf(“%f”, MEDIA);

passo 7: scanf(“%f%f%f”,&N1,&N2,&N3);

passo 8: MEDIA = (N1 + N2 + N3) / 3;

passo 9: printf(“%f”, MEDIA);

passo 10: scanf(“%f%f%f”,&N1,&N2,&N3);

passo 11: MEDIA = (N1 + N2 + N3) / 3;

passo 12: printf(“%f”, MEDIA);

Comandos Repetitivos (laços)

- E Para 10 alunos?
- **QUANTOS ALUNOS TEM, EM MÉDIA, UMA TURMA?**
  - 40, 50?
- Que tamanho ficaria esse programa para uma turma com 50 alunos?

Comandos Repetitivos (laços)

- Observe que os 3 primeiros passos são repetidos tantas vezes quanto solicitado.
- Uma forma de escrever o mesmo programa sem repetir várias vezes os mesmos passos é usando uma estrutura de repetição ou laço:
- Por exemplo:

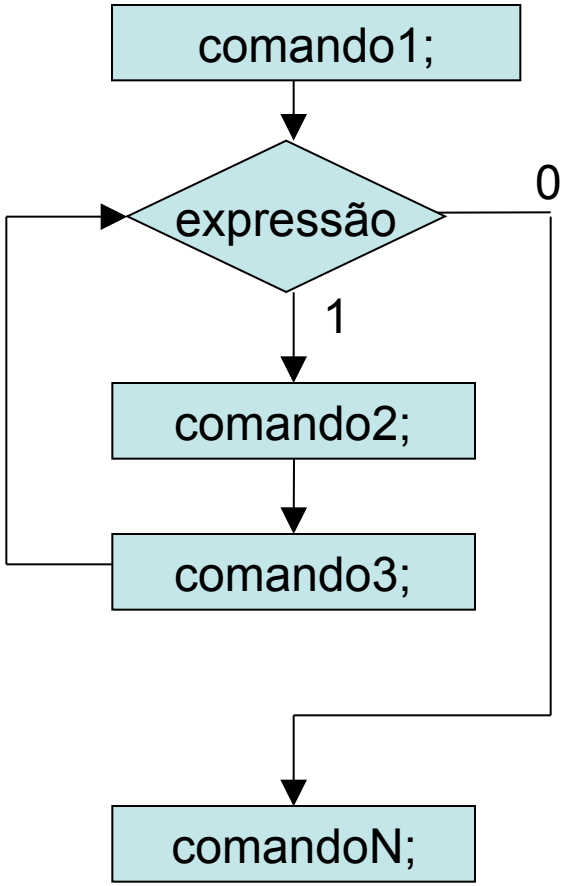
```
ContaAlunos =0
REPITA
        passo 1: leia N1, N2, N3
        passo 2: MEDIA ← (N1 + N2 + N3) / 3
        passo 3: imprima MEDIA
        passo 4: ContaAlunos = ContaAlunos +1
ATÉ QUE ContaAlunos=50
```

Comandos Repetitivos (laços)

- Desejamos que um bloco de comandos seja executado repetidamente até que determinada condição seja satisfeita.
- Em C existem 3 tipos principais de laços:
  - Laço **while** **(teste no início)**
  - Laço **do-while** **(teste no final)**
  - Laço **for** **(variável de controle)**

Laço “**while**”

- É uma instrução de repetição, onde a expressão lógica é testada antes de executar o bloco de comandos.




```
comando1;
while(expressão){
    // bloco de comandos.
    comando2;
    comando3;
    :
}
comandoN;
```

Laço "while" (Exemplo)

- **Problema:** somar **n** valores lidos da entrada padrão.

```
#include <stdio.h>
int main() {
    int n,num,i,soma=0;
    printf("Entre com a quantidade de números a
somar: ");
    scanf("%i",&n);
    i=0;
    while (i<n) {
        printf("Entre com um valor: ");
        scanf("%i",&num);
        soma+=num;
        i++; // i=i+1
    }
    printf("Soma: %i",soma);
    return 0;
}
```

” (Exemplo)

- **Problem** s lidos da entrada padrão.

variável de controle

```
#include <st   b.h>
int main() {
    int n,num,i,soma=0;
    printf("Entre com a quantidade de números a
somar: ");
    scanf("%i",&n);
    i=0;
    while (i<n) {
        printf("Entre com um valor: ");
        scanf("%i",&num);
        soma+=num;
        i++; // i=i+1
    }
    printf("Soma: %i",soma);
    return 0;
}
```

inicialização

” (Exemplo)

- **Problem** s lidos da entrada padrão.

variável de controle

```
#include <st   o.h>
int main() {
    int n,num,i,soma=0;
    printf("Entre com a quantidade de números a
somar: ");
    scanf("%i",&n);
    i=0;
    while (i<n) {
        printf("Entre com um valor: ");
        scanf("%i",&num);
        soma+=num;
        i++; // i=i+1
    }
    printf("Soma: %i",soma);
    return 0;
}
```

expressão

” (Exemplo)

- **Problem** s lidos da entrada padrão.

variável de controle

```
#include <st  o.h>
int main() {
    int n,num,i,soma=0;
    printf("Entre com a quantidade de números a
somar: ");
    scanf("%i",&n);
    i=0;
    while (i<n) {
        printf("Entre com um valor: ");
        scanf("%i",&num);
        soma+=num;
        i++; // i=i+1
    }
    printf("Soma: %i",soma);
    return 0;
}
```

atualização

Laço "**while**" (Exemplo)

- **Problema:** Calcular o fatorial de um número.

```
int num, fat;

printf("Digite um número: ");
scanf("%d",&num);

fat = 1;
while(num>1){
   fat *= num;    // fat = fat * num;
   num--;
}
printf("fatorial: %d\n",fat);
```

Comandos Repetitivos (laços)

- **Os comados break e continue:**
  - Podem ser usados no corpo de qualquer estrutura de laço C.
  - O **break** causa a saída imediata do laço e o controle passa para a próxima instrução após o laço.
  - O **continue** força a próxima iteração do laço e pula o código que estiver abaixo.

```
#include <stdio.h>
int main(){
    int soma=0,num,n;
    n=1;
    while (n<=5) {
        scanf("%i",&num);
        if (num<0) continue; ;  // filtrar números negativos
        soma+=num; // soma = soma + num
        n++;
    }
    printf("\nA soma é %i",soma);
    return 0;
}
```

Laço "while" (Exemplo)

```
// prog.para calcular a media das notas de 10 alunos"
#include <stdio.h>
main()
{
    int cont=0;
    float n1, n2, media;
    while (cont < 10)
    {
      scanf("%f%f",&n1,&n2);
      media = (n1+n2)/2);
      printf("media do aluno %d: %f", cont,media)
      cont++;
    }
}
```

Iniciando o contador

Condição

Atualização do contador

Uso de {} porque tem mais de uma linha de comando dentro do while

Comandos Repetitivos (laços)

- **Laço infinito:**

```
#include <stdio.h>

int main(){
  int i=0;

  while(1){
     printf("%d\n", i);
     i++;
  }
  return 0;
}
```

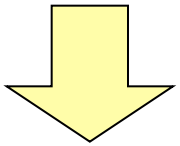
Laços aninhados

$$S = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{i^2 * j}{3^i (j * 3^i + i * 3^j)}$$

```
#include <stdio.h>
#include <math.h>
int main(){
    int i,j,m,n;
    float S=0.0,pi,pj,v;
    scanf("%d %d",&n,&m);
    i=1;
    while (i<=n) {
        j=1;
        while (j<=m) {
            pi = powf(3.0,  i);
            pj = powf(3.0,  j);
            v = (float) i * i * j;
            v /= pi*(j*pi + i*pj);
            S += v;      // S = S + v;
            j++;
        }
        i++;
    }
    printf("Soma: %f\n",S);
    return 0;
}
```

Laços aninhados

$$S = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{i^2 * j}{3^i (j * 3^i + i * 3^j)}$$



$$S = \sum_{i=1}^{n} \left(\frac{i^2}{3^i} \sum_{j=1}^{m} \frac{j}{j * 3^i + i * 3^j} \right)$$

```
#include <stdio.h>
#include <math.h>
int main(){
    int i,j,m,n;
    float S=0.0,Si,pi,pj;
    scanf("%d %d",&n,&m);
    i=1;
    while (i<=n) {
        Si = 0.0;
        pi = powf(3.0,  i);
        j=1;
        while (j<=m) {
            pj = powf(3.0,  j);
            Si += j / (j*pi + i*pj);
            j++;
        }
        S += (Si*i*i) / pi;
        i++;
    }
    printf("Soma: %f\n",S);
    return 0;
}
```

Laços aninhados

- **Problema:**
  - Use o comando **while** para imprimir as seguintes sequências de números:

(1, 1 2 3 4 5 6 7 8 9 10)

(2, 1 2 3 4 5 6 7 8 9 10)

(3, 1 2 3 4 5 6 7 8 9 10)

(4, 1 2 3 4 5 6 7 8 9 10)

...

(10, 1 2 3 4 5 6 7 8 9 10)

e assim sucessivamente, até que o primeiro número (antes da vírgula), também chegue a 10.

Laços aninhados

- **Problema:** sequência de números

```
#include <stdio.h>
int main(){
    int i,n=1;
    while (n<=10) {
        printf("\n ( %d, ",n);
        i=1;
        while (i<=10) {
            printf("%d ",i);
            i++;
        }
        printf(")");
        n++;
    }
    return 0;
}
```

Laços aninhados

- **Problema:**
  - Use o comando **while** para mostrar uma pirâmide semelhante a abaixo, sendo que o maior valor da pirâmide é definido pelo usuário. Ex: n=9

```
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Laços aninhados

- **Problema:** pirâmide

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

Laços aninhados

- **Problema:**
  - Use o comando **while** para mostrar uma tabuada semelhante a abaixo:

```
TABUADA DO 2  TABUADA DO 3  TABUADA DO 4  TABUADA DO 5
 2 x 1 =  2    3 x 1 =  3    4 x 1 =  4    5 x 1 =  5
 2 x 2 =  4    3 x 2 =  6    4 x 2 =  8    5 x 2 = 10
 2 x 3 =  6    3 x 3 =  9    4 x 3 = 12    5 x 3 = 15
 2 x 4 =  8    3 x 4 = 12    4 x 4 = 16    5 x 4 = 20
 2 x 5 = 10    3 x 5 = 15    4 x 5 = 20    5 x 5 = 25
 2 x 6 = 12    3 x 6 = 18    4 x 6 = 24    5 x 6 = 30
 2 x 7 = 14    3 x 7 = 21    4 x 7 = 28    5 x 7 = 35
 2 x 8 = 16    3 x 8 = 24    4 x 8 = 32    5 x 8 = 40
 2 x 9 = 18    3 x 9 = 27    4 x 9 = 36    5 x 9 = 45

TABUADA DO 6  TABUADA DO 7  TABUADA DO 8  TABUADA DO 9
 6 x 1 =  6    7 x 1 =  7    8 x 1 =  8    9 x 1 =  9
 6 x 2 = 12    7 x 2 = 14    8 x 2 = 16    9 x 2 = 18
 6 x 3 = 18    7 x 3 = 21    8 x 3 = 24    9 x 3 = 27
 6 x 4 = 24    7 x 4 = 28    8 x 4 = 32    9 x 4 = 36
 6 x 5 = 30    7 x 5 = 35    8 x 5 = 40    9 x 5 = 45
 6 x 6 = 36    7 x 6 = 42    8 x 6 = 48    9 x 6 = 54
 6 x 7 = 42    7 x 7 = 49    8 x 7 = 56    9 x 7 = 63
 6 x 8 = 48    7 x 8 = 56    8 x 8 = 64    9 x 8 = 72
 6 x 9 = 54    7 x 9 = 63    8 x 9 = 72    9 x 9 = 81
```

Laços aninhados

- **Problema:** tabuada

```
#include <stdio.h>
int main(){
    int i,j,k,t;
    k=0;
    while (k<=1){
        printf("\n");
        t=1;
        while (t<5){
            printf(" TABUADA DO %d ",t+4*k+1);
             t++;
        }
        printf("\n");
        i=1;
        while (i<=9){
            j=2+4*k;
            while (j<=5+4*k) {
                printf("  %d x %d = %2d  ", j, i, j*i);
                j++;
            }
            printf("\n");
            i++;
        }
         k++;
    }
    return 0;
}
```

Laços aninhados

- **Problema:**
  - Use o comando **while** para mostrar a figura abaixo:

```
. # # # # # # # # # .
. . # # # # # # # . .
. . . # # # # # . . .
. . . . # # # . . . .
. . . . . # . . . . .
. . . . . . . . . . .
. . . . . # . . . . .
. . . . # # # . . . .
. . . # # # # # . . .
. . # # # # # # # . .
. # # # # # # # # # .
```

Laços aninhados

- **Problema:** pontos em 2D

```
#include <stdio.h>
int main(){
    int x,y,expr;
    y=10;
    while (y>=0){
        x=0;
        while (x<=10) {
            expr = (x<y && y>10-x) || (x>y && y<10-x);
            if(expr)   // expr==1
                printf(" # ");
            else       // expr==0
                printf(" . ");
            x++;
        }
        printf("\n");
        y--;
    }
    return 0;
}
```