Linguagem de Programação II

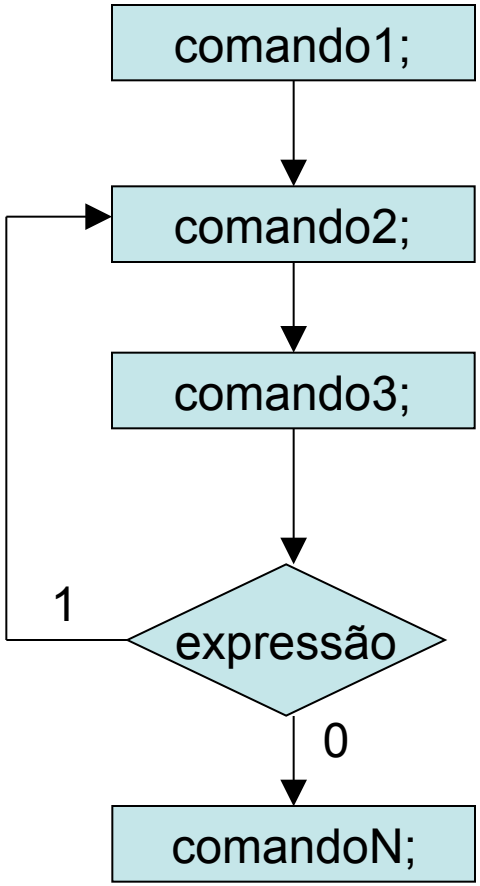
Prof. Mario Bessa

Aula 7

http://mariobessa.info

Laço “do-while”

- É uma instrução de repetição, onde a condição de interrupção (expressão lógica) é testada após executar o bloco de comandos.




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

Comandos Repetitivos (laços)

- **Equivalências:**

```
do{
    comando1;
    comando2;
    …
}while(exp1);
```

```
while(exp1){
    comando1;
    comando2;
    …
}
```

Comandos Repetitivos (laços)

- **Os comados break e continue:**
  - Podem ser usados no corpo de qualquer estrutura de laço em 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.

Laço “do-while” (Exemplo)

```
float num;
int op;
 do{
    printf("\nDigite um número: ");
    scanf("%f",&num);
    printf("1) raiz quadrada.\n");
    printf("2) log na base 10.\n");
    printf("3) tangente.\n");
    printf("4) Sair.\n");
    printf("Escolha uma operação: ");
    scanf("%d",&op);
    if(op==1)  printf(”%f\n",sqrt(num));
    else if(op==2)  printf(”%f\n",log10(num));
    else if(op==3)  printf(”%f\n",tan(num));
 }while(op!=4);
```

Comandos Repetitivos (laços)

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

Laços aninhados

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

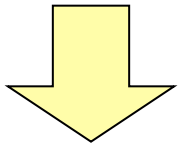
Laços aninhados

```
#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;
    do {
        j=1;
        do {
            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++;
        } while (j<=m);
        i++;
    } while (i<=n);
    printf("Soma: %f\n",S);
    return 0;
}
```

```
#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

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



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

Laços aninhados

```
#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;
    do {
        Si = 0.0;
        pi = powf(3.0,  i);
        j=1;
        do {
            pj = powf(3.0,  j);
            Si += j / (j*pi + i*pj);
            j++;
        } while (j<=m);
        S += (Si*i*i) / pi;
        i++;
    } while (i<=n);
    printf("Soma: %f\n",S);
    return 0;
}
```

```
#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 **do 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;
    do {
        printf("\n ( %d, ",n);
        i=1;
        do {
            printf("%d ",i);
            i++;
        } while (i<=10);
        printf(")");
        n++;
    } while (n<=10);
    return 0;
}
```

```
#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 **do 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;
    do {
        j=i;
        do {
            printf(" %d ", j);
            j--;
        } while (j!=0);
        printf("\n");
        i--;
    } while (i!=0);
    return 0;
}
```

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

```
#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 **do while** para mostrar a figura abaixo:

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

Laços aninhados

- **Problema:** pontos em 2D

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

```
#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;
}
```

- **Problema:**
  - Número de Armstrong: é um número de n dígitos que é igual a soma de cada um dos seus dígitos elevado a n-ésima potência.
  - Exemplos:
    - $153=1^3 + 5^3 + 3^3$
    - $370=3^3+7^3+0^3$
    - $371=3^3+7^3+1^3$
    - $407=4^3+0^3+7^3$
  - Elabore um programa em C para ler um inteiro e dizer se é ou não Armstrong.
  - Altere esse programa para imprimir todos os números Armstrong entre 1 e 9999999.

```c
#include <stdio.h>
#include <math.h>
int main(){
    int num,soma=0,temp,resto,n=0;
    printf("Enter an integer\n");
    scanf("%d",&num);
    temp = num;
    while( temp != 0 ){
        n++;
        temp = temp/10;
    }
    temp = num;
    while( temp != 0 ){
        resto = temp%10;
        soma = soma + pow(resto,n);
        temp = temp/10;
    }
    if ( num == soma )
        printf("É armstrong.\n");
    else
        printf("Não é armstrong.\n");
    return 0;
}
```

```c
#include <stdio.h>
#include <math.h>
int main(){
    int num=1,soma,temp,resto,n;
    while (num<9999999) {
        n=0;
        soma=0;
        temp = num;
        while(temp != 0){
            n++;
            temp = temp/10;
        }
        temp = num;
        while(temp != 0){
            resto = temp%10;
            soma = soma + pow(resto,n);
            temp = temp/10;
        }
        if ( num == soma )printf("%i ",num);
        num++;
    }
    return 0;
}
```