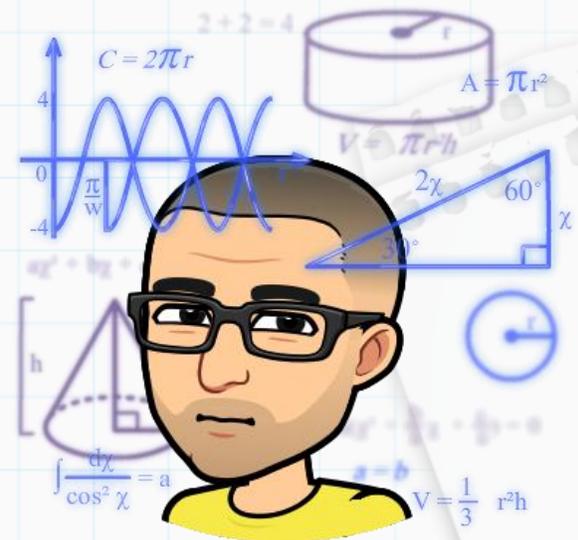
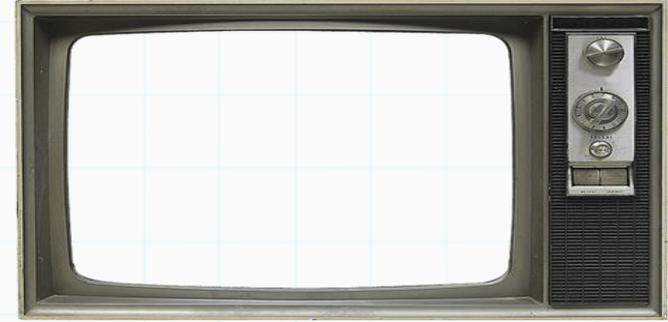


# Programação Estruturada

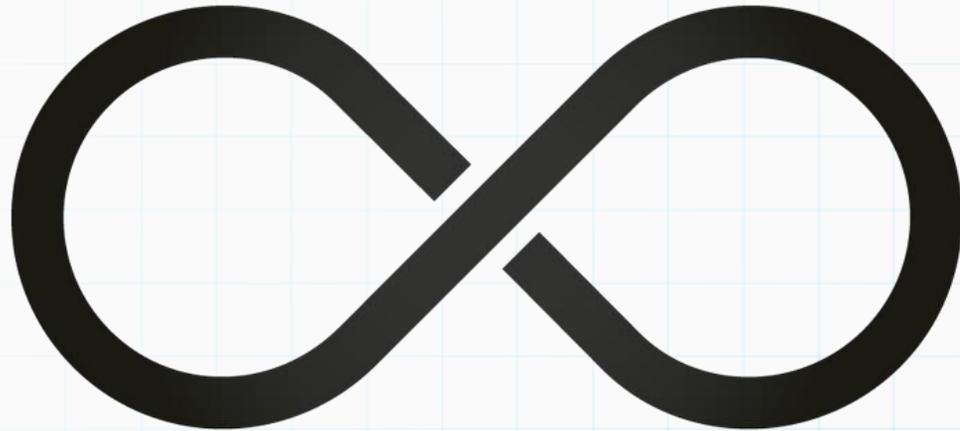
Professor : Yuri Frota

yuri@ic.uff.br



# Repetição

Um comando de repetição é aquele que permite repetir um determinado bloco de comandos. Existem dois tipos de repetição: as condicionais e as contáveis.



# Repetição Contável

Comando For:

Executa o bloco de instruções um número fixo de vezes

Portugol

...

para VAR variando de VALOR  
INICIAL enquanto o teste  
de PARADA valer, repita, e  
a cada iteração incrementa  
em INC

INSTRUÇÃO 1

INSTRUÇÃO 2

...

INSTRUÇÃO N

...



# Repetição Contável

Comando For:

Executa o bloco de instruções um número fixo de vezes

Portugol

```
...  
para VAR variando de VALOR  
INICIAL enquanto o teste  
de PARADA valer, repita, e  
a cada iteração incrementa  
em INC
```

```
INSTRUÇÃO 1
```

```
INSTRUÇÃO 2
```

```
...
```

```
INSTRUÇÃO N
```

```
...
```

C

```
...  
for (<INICIO>;<PARADA>;<INC>)  
{
```

```
INSTRUÇÃO 1;
```

```
INSTRUÇÃO 2;
```

```
...
```

```
INSTRUÇÃO N;
```

```
}
```

```
...
```

Parâmetros entre “( )” separados por “;”



X é iniciada em <início>, e a cada iteração do laço, incrementa X em <inc>, e executa enquanto teste <parada> for verdadeiro

# Repetição Contável

Somatório de 1 a 10:

```
int main (void) {  
    int soma = 0;  
    int i;  
    for (i=1; i <= 10; i++)  
        soma += i;  
    printf("soma = %d", soma);  
    return 0;  
}
```



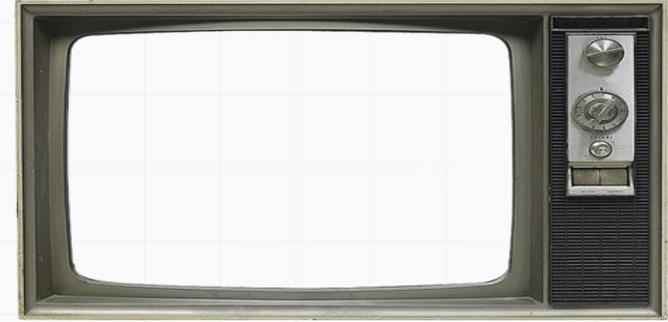
# Repetição Contável

Somatório de 1 a 10:

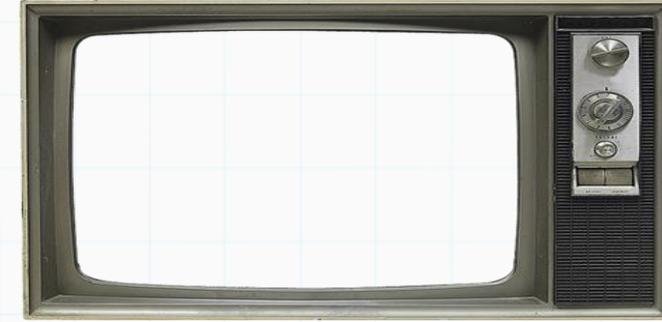
```
int main (void) {
    int soma = 0;
    int i;
    for (i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```

Podemos iniciar vários valores de uma vez, separados por “,”

```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```



# Repetição Contável



Somatório de 1 a 10:

```
int main (void) {
    int soma = 0;
    int i;
    for (i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```

Podemos realizar o acúmulo da soma na expressão da iteração

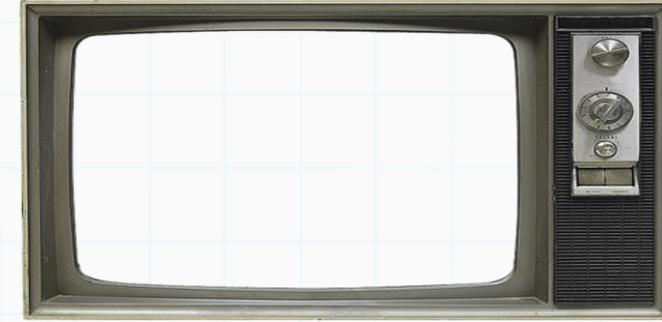
```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; soma +=i++);
    printf("soma = %d", soma);
    return 0;
}
```

Podemos iniciar vários valores de uma vez, separados por “,”

```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```



# Repetição Contável



Somatório de 1 a 10:

```
int main (void) {
    int soma = 0;
    int i;
    for (i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```

Podemos realizar o acúmulo da soma na expressão da iteração

```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; soma +=i++);
    printf("soma = %d", soma);
    return 0;
}
```

Podemos iniciar vários valores de uma vez, separados por “,”

```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; i++)
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```

Podemos fazer laços infinitos ( for(;;) )

```
int main (void) {
    int soma;
    int i;
    for (soma=0, i=1; i <= 10; )
        soma += i;
    printf("soma = %d", soma);
    return 0;
}
```

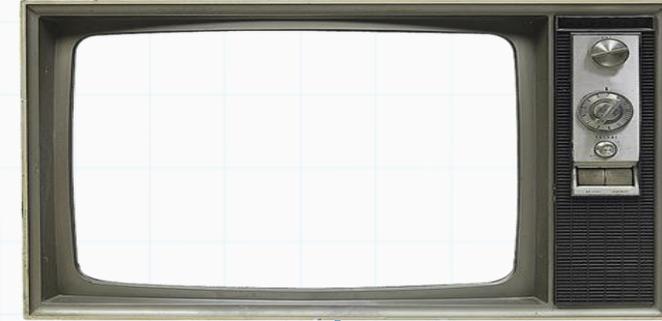
# Repetição Contável

Comandos que alteram o fluxo da repetição:

**break:** encerra o laço imediatamente, mesmo se a condição de término não tiver sido alcançada.

```
int main()
{
    for (int i ; i <10 ; i++)
    {
        if (i==5)
            break;
        printf("%d\n",i);
    }
    printf("fim\n");
}
```

```
0
1
2
3
4
fim
```



# Repetição Contável

Comandos que alteram o fluxo da repetição:

**break:** encerra o laço imediatamente, mesmo se a condição de término não tiver sido alcançada.

Quebra apenas o mais interno

```
int main()
{
    for (int i=0; i<10; i++)
    {
        if (i==5)
            break;
        printf("%d\n",i);
    }
    printf("fim\n");
}
```

```
0
1
2
3
4
fim
```

```
int main()
{
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<10; j++)
        {
            if (j==3)
                break;
            printf("%d %d\n",i,j);
        }
    }
    return 0;
}
```

```
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
```



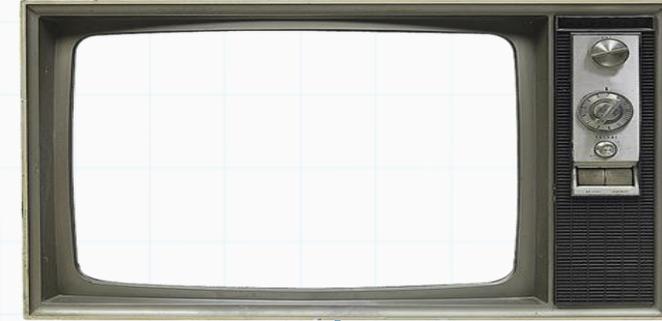
# Repetição Contável

Comandos que alteram o fluxo da repetição:

continue: passa imediatamente para a próxima iteração do laço.

```
int main()  
{  
    for (int i=0; i<10; i++)  
    {  
        if (i==5)  
            continue;  
        printf("%d\n", i);  
    }  
    printf("fim\n");  
}
```

```
0  
1  
2  
3  
4  
6  
7  
8  
9  
fim
```



# Repetição Contável

Comandos que alteram o fluxo da repetição:

continue: passa imediatamente para a próxima iteração do laço.

```
int main()
{
    for (int i=0; i<10; i++)
    {
        if (i==5)
            continue;
        printf("%d\n", i);
    }
    printf("fim\n");
}
```

```
0
1
2
3
4
6
7
8
9
fim
```

continua apenas o mais interno

```
int main()
{
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<10; j++)
        {
            if (j==3)
                continue;
            printf("%d %d\n", i, j);
        }
    }
    return 0;
}
```

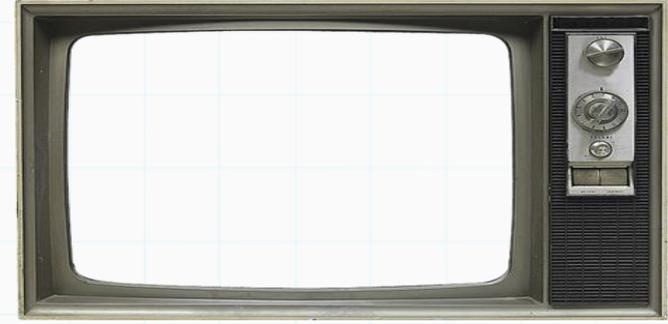


```
0 1 1 6
0 2 1 7
0 4 1 8
0 5 1 9
0 6 2 0
0 7 2 1
0 8 2 2
0 9 2 4
1 0 2 5
1 1 2 6
1 2 2 7
1 4 2 8
1 5 2 9
```

# Repetição Condicional

Comando While:

Executa o bloco de instruções enquanto a condição for verdadeira



Portugol

...

enquanto **CONDIÇÃO**

faça

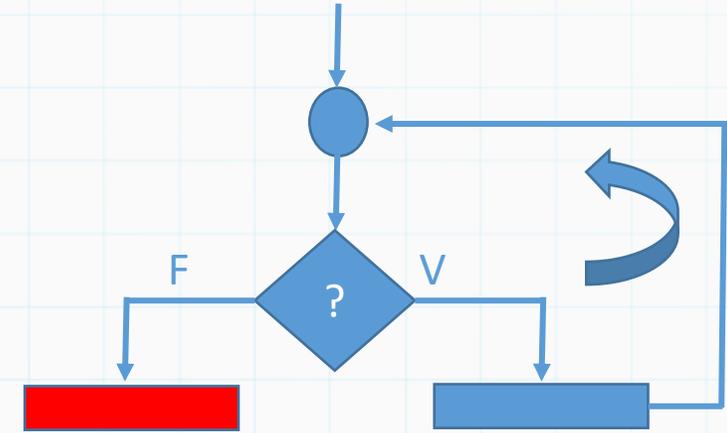
**INSTRUÇÃO 1**

**INSTRUÇÃO 2**

...

**INSTRUÇÃO N**

...



# Repetição Condicional

Comando While:

Executa o bloco de instruções enquanto a condição for verdadeira

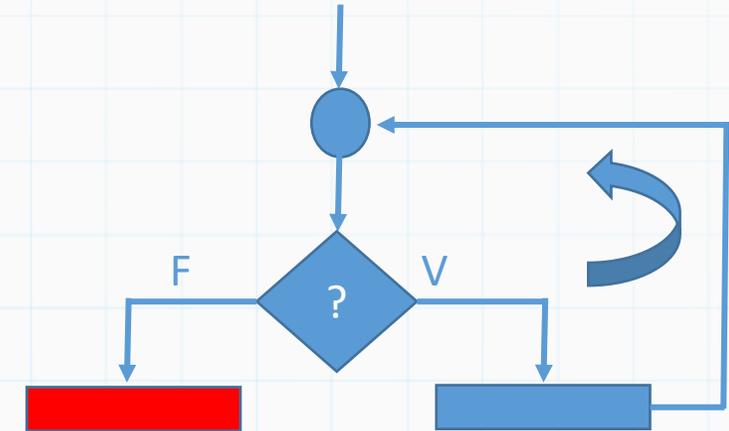


Portugol

```
...
enquanto CONDIÇÃO
faça
    INSTRUÇÃO 1
    INSTRUÇÃO 2
    ...
    INSTRUÇÃO N
...
```

C

```
...
while (CONDIÇÃO)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}
...
while (CONDIÇÃO)
    INSTRUÇÃO 1;
```

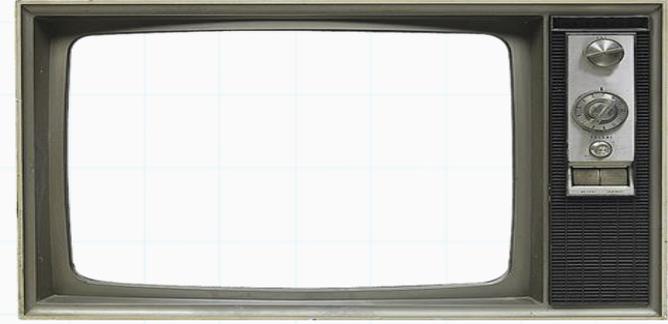


- Condição entre “( )”
- Se tiver apenas 1 instrução não precisa de “{ }”
- Quando a condição se torna falsa, a **próxima instrução** após o bloco do **while** é executada.
- Se a condição do **while** for falsa desde o início, o bloco de instruções **nunca é executado**

# Repetição Condicional

Somatório de 1 a 10:

```
int main (void) {  
    int soma=0;  
    int i=1;  
    while (i <= 10)  
    {  
        soma += i;  
        i++;  
    }  
    printf("soma = %d", soma);  
    return 0;  
}
```



# Repetição Condicional

Comando Do-While:

Executa o bloco de instruções enquanto a condição for verdadeira

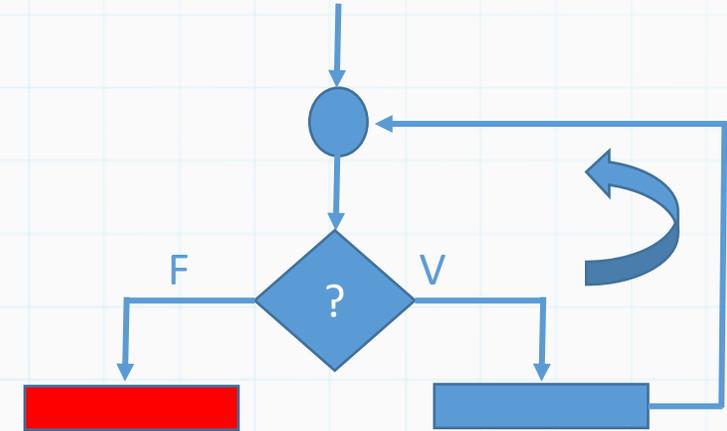


Portugol

```
...  
faça  
    INSTRUÇÃO 1  
    INSTRUÇÃO 2  
    ...  
    INSTRUÇÃO N  
enquanto CONDIÇÃO  
...
```

C

```
...  
do  
{  
    INSTRUÇÃO 1;  
    INSTRUÇÃO 2;  
    ...  
    INSTRUÇÃO N;  
}  
while (CONDIÇÃO);  
...  
do  
    INSTRUÇÃO 1;  
while (CONDIÇÃO);
```

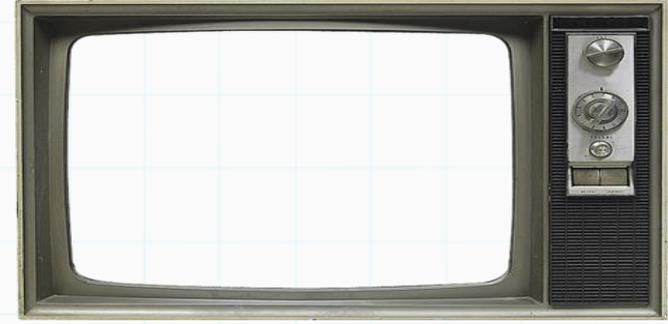


- Condição entre “()”
- Se tiver apenas 1 instrução não precisa de “{}”
- Mesma coisa do While, mas o teste agora é no fim da estrutura.
- Executa pelo menos uma vez

# Repetição Condicional

Somatório de 1 a 10:

```
int main (void) {
    int soma=0;
    int i=1;
    do
    {
        soma += i;
        i++;
    }
    while (i <= 10);
    printf("soma = %d", soma);
    return 0;
}
```



# Repetição



Fura Olho: O que será escrito ?

```
void main()
{
    double k = 0;
    for (k = 0.0; k < 3.0; k++);
    printf("%lf", k);
}
```

```
void main()
{
    int i = 0;
    do
    {
        printf("Hello");
    } while (i != 0);
}
```

```
void main()
{
    int k;
    for (k = -3; k < -5; k++)
        printf("Hello");
}
```

```
void main()
]{
    int i = 0;

    while (i < 5)
    {
        i++;
        printf("pika\n");
        while (i < 3)
        {
            i++;
            printf("chu\n");
        }
    }
}
```

# Número Aleatório



Como gerar um número aleatório entre 0 e 20 ?

```
#include <stdio.h>

#include <stdlib.h>
#include <time.h>

int main()
{
    // Variável de tempo
    time_t t;

    //inicializa semente aleatória
    srand( (int) time(&t));

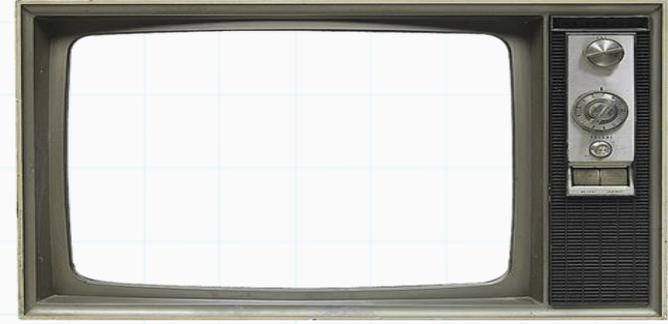
    //gera número entre 0 e 20
    int num = rand() % 21;

    printf("num = %d\n", num);
}
```

- t é uma variável de tempo
- **time()** retorna (em segundos) o tempo desde 00:00:00 UTC, January 1, 1970. Passamos um ponteiro para a variável que irá atualizar o valor dela.



# Número Aleatório



Como gerar um número aleatório entre 0 e 20 ?

```
#include <stdio.h>

#include <stdlib.h>
#include <time.h>

int main()
{
    // Variável de tempo
    time_t t;

    //inicializa semente aleatória
    srand( (int) time(&t));

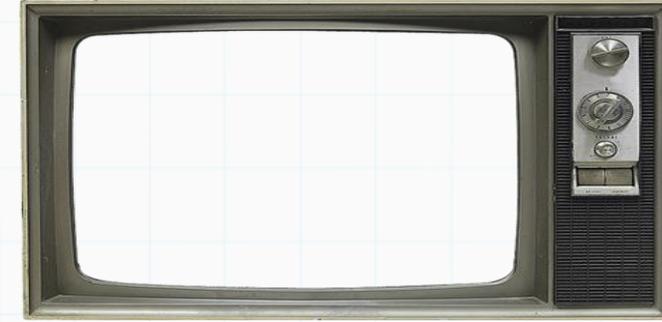
    //gera número entre 0 e 20
    int num = rand() % 21;

    printf("num = %d\n", num);
}
```

- t é uma variável de tempo
- **time()** retorna (em segundos) o tempo desde 00:00:00 UTC, January 1, 1970. Passamos um ponteiro para a variável que irá atualizar o valor dela.
- **srand()** inicia a semente aleatória.



# Número Aleatório



Como gerar um número aleatório entre 0 e 20 ?

```
#include <stdio.h>

#include <stdlib.h>
#include <time.h>

int main()
{
    // Variável de tempo
    time_t t;

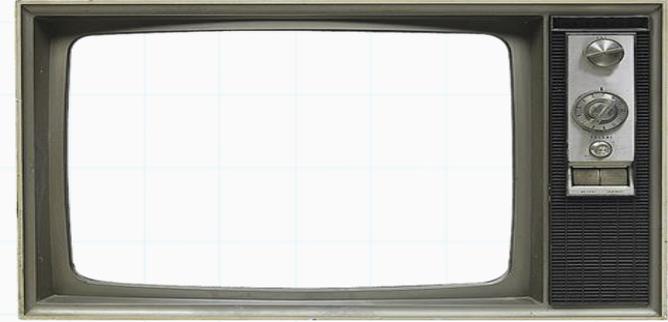
    //inicializa semente aleatória
    srand( (int) time(&t));

    //gera número entre 0 e 20
    int num = rand() % 21;

    printf("num = %d\n", num);
}
```

- t é uma variável de tempo
- **time()** retorna (em segundos) o tempo desde 00:00:00 UTC, January 1, 1970. Passamos um ponteiro para a variável que irá atualizar o valor dela.
- **srand()** inicia a semente aleatória.
- **rand()** gera um número pseudo-aleatório entre 0 e um limite RAND\_MAX, se semente não iniciada, por padrão é 1
- **time()** pertence a **time.h**, **srand()** e **rand()** pertencem a **stdlib.h**

# Número Aleatório



## Simulando um dado

```
#include <stdio.h>

#include <stdlib.h>
#include <time.h>

int main()
{
    // Variável de tempo
    time_t t;

    //inicializa semente aleatória
    srand( (int) time(&t));

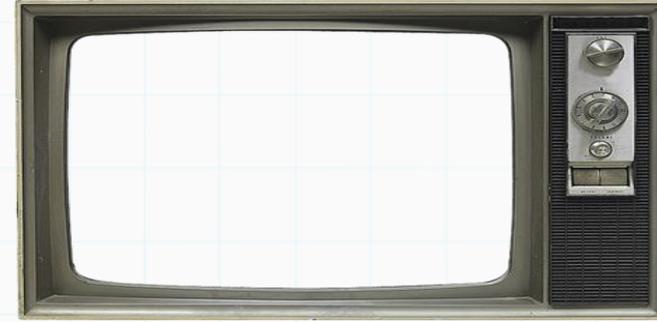
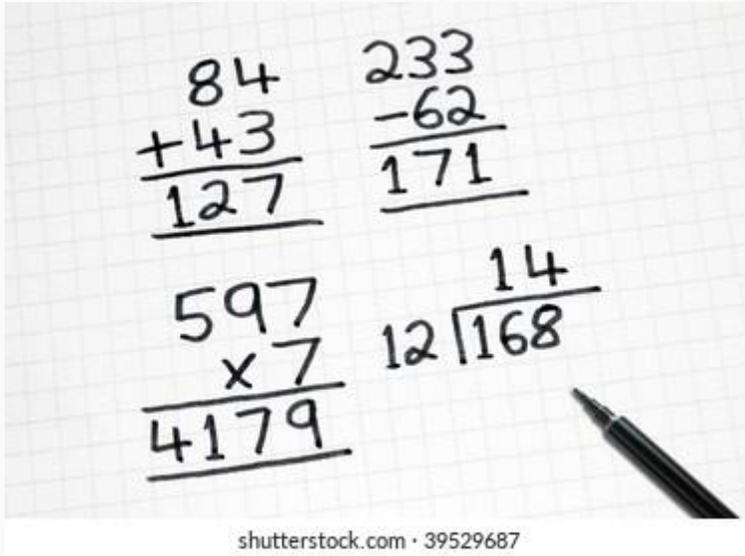
    //gera número entre 1 e 6
    int dado = (rand() % 6) + 1;

    printf("dado = %d\n", dado);
}
```

- t é uma variável de tempo
- **time()** retorna (em segundos) o tempo desde 00:00:00 UTC, January 1, 1970. Passamos um ponteiro para a variável que irá atualizar o valor dela.
- **srand()** inicia a semente aleatória.
- **rand()** gera um número pseudo-aleatório entre 0 e um limite RAND\_MAX, se semente não iniciada, por padrão é 1
- **time()** pertence a **time.h**, **srand()** e **rand()** pertencem a **stdlib.h**

# Exercícios

1) Somatórios: Fazer um programa que leia N números naturais positivos e que escreva o somatório dos números pares e a média dos múltiplos de 3.



```
for (<INICIO>;<PARADA>;<INC>)  
{  
    INSTRUÇÃO 1;  
    INSTRUÇÃO 2;  
    ...  
    INSTRUÇÃO N;  
}
```

```
while (CONDIÇÃO)  
{  
    INSTRUÇÃO 1;  
    INSTRUÇÃO 2;  
    ...  
    INSTRUÇÃO N;  
}
```

&& -> e lógico  
|| -> ou lógico

```
printf("a = %d", a);  
scanf("%d", &n);
```

```
int main() {
```

```
int n, num, n3=0;  
float media=0;
```

```
printf("n : ");  
scanf("%d", &n);
```

```
for (int i = 1; i <= n; ++i)  
{
```

```
printf("num: ");  
scanf("%d", &num);
```

```
if (num % 2 == 0)  
{
```

```
int soma = 0;  
for (int j = 1; j <= num; ++j)  
soma += j;  
printf(" soma = %d\n", soma);  
}
```

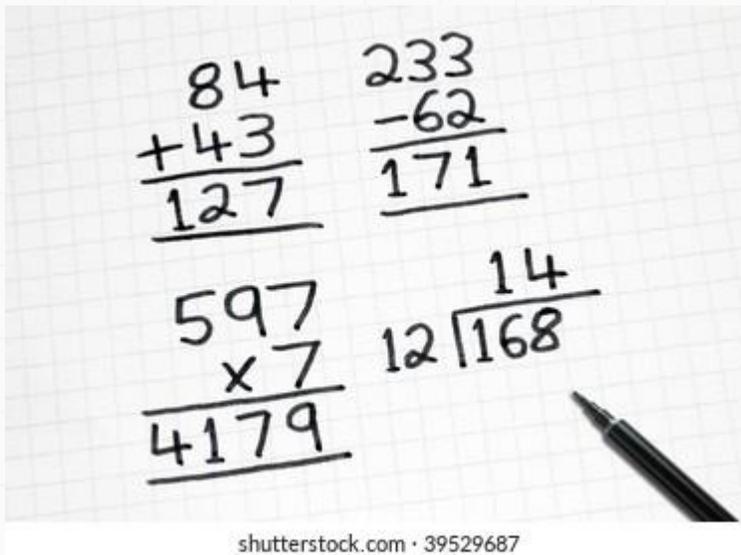
```
if (num % 3 == 0)  
{
```

```
media += num;  
n3 += 1;  
}
```

```
printf(" media = %.2f\n", media/n3);
```

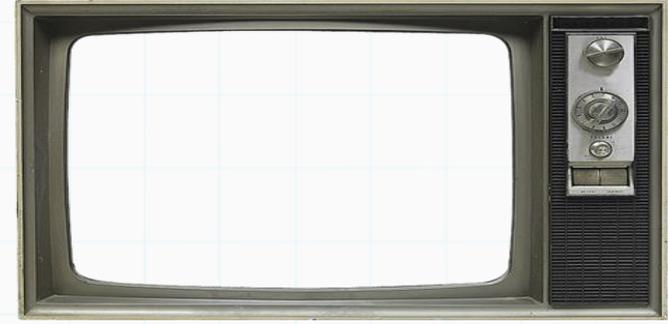
```
return 0;
```

# Exercícios



# Exercícios

2) Salários: Calcular o salário atual de um funcionário sabendo que ele foi contratado por 1000 reais em 1995 e que no ano seguinte recebeu um aumento de 0.1%, A partir daí, em cada ano o funcionário recebeu um percentual de aumento do dobro do ano anterior.



```
for (<INICIO>;<PARADA>;<INC>)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}
```

```
while (CONDIÇÃO)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}
```

```
&& -> e lógico
|| -> ou lógico
```

```
printf("a = %d",a);
scanf("%d",&n);
```

# Exercícios



```
#include <stdio.h>

int main() {

    int    ano;
    double salario;
    float  percentual = 0.1;

    salario = 1000;

    for(ano = 1996; ano <= 2024; ano ++){
        salario    = salario + (salario*(percentual/100));
        percentual  = percentual * 2;

        printf("Salario em %d: R$ %.2lf percentual=%f\n", ano, salario, percentual);
    }

    return 0;
}
```

# Exercícios

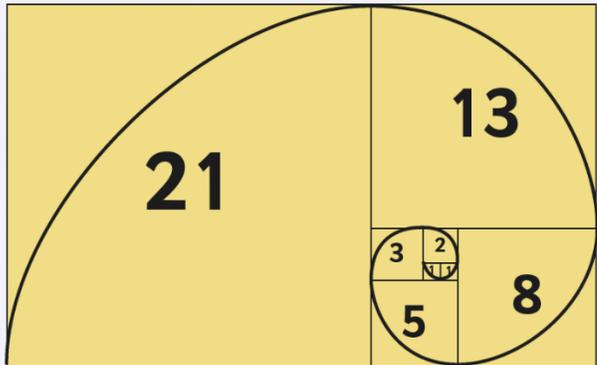
3) Fibonacci: Fazer um programa que escreva a série de Fibonacci, cujo último termo seja menor ou igual a N (inteiro). Obrigar o usuário a digitar um valor positivo para N.

Sabendo que a série começa com 0 e 1, e próximo é a soma dos dois anteriores:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ...



Fibonacci Spirals

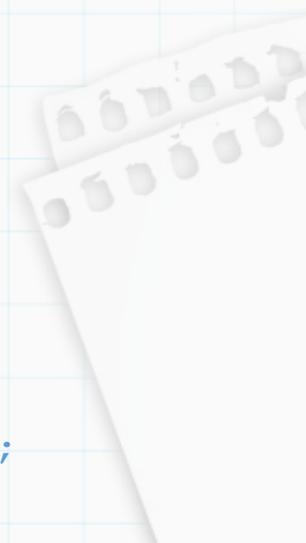


```
for (<INICIO>;<PARADA>;<INC>)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}

while (CONDIÇÃO)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}

&& -> e lógico
|| -> ou lógico

printf("a = %d", a);
scanf("%d", &n);
```



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

```
int main() {
```

```
    int i, n=-1;
```

```
    int t1 = 0, t2 = 1;
```

```
    int prox = t1 + t2;
```

```
    while (n < 0)
```

```
    {  
        printf("n: ");  
        scanf("%d", &n);  
    }
```

```
    if (n >=1)  
        printf("%d, ", t1);
```

```
    if (n >=2)  
        printf("%d, ", t2);
```

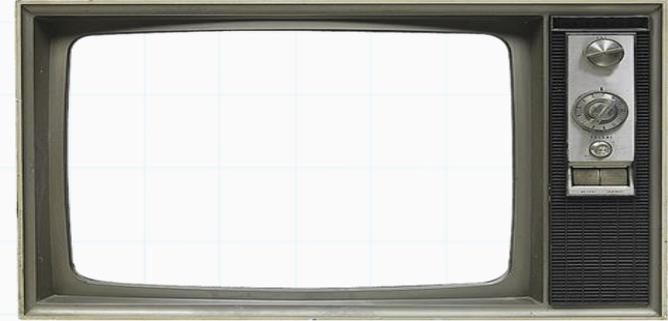
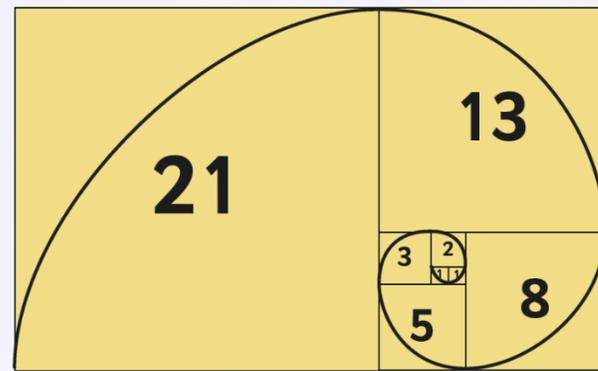
```
    for (i = 3; i <= n; ++i)  
    {
```

```
        printf("%d, ", prox);  
        t1 = t2;  
        t2 = prox;  
        prox = t1 + t2;  
    }
```

```
    return 0;
```

# Exercícios

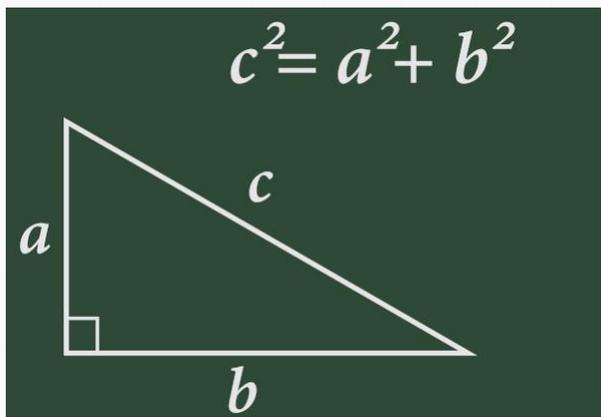
## Fibonacci Spirals



# Exercícios

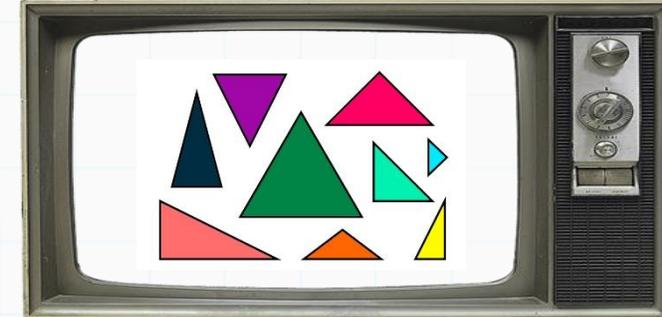
4) Triângulos: Dado um número inteiro positivo  $n$ , determinar todos os inteiros entre 1 e  $n$  que são comprimento da hipotenusa de um triângulo retângulo com catetos inteiros.

Sabendo que:



Exemplo:

```
Digite o comprimento maximo da hipotenusa: 20
hipotenusa = 5, catetos 3 e 4
hipotenusa = 10, catetos 6 e 8
hipotenusa = 13, catetos 5 e 12
hipotenusa = 15, catetos 9 e 12
hipotenusa = 17, catetos 8 e 15
hipotenusa = 20, catetos 12 e 16
```



```
for (<INICIO>;<PARADA>;<INC>)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}

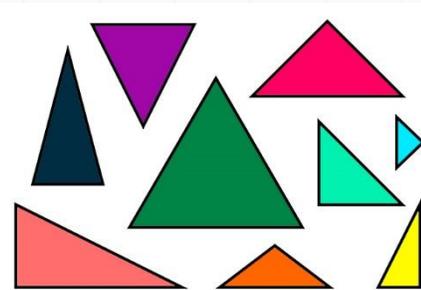
while (CONDIÇÃO)
{
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}

&& -> e lógico
|| -> ou lógico

printf("a = %d", a);
scanf("%d", &n);
```



## Exercícios



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

```
int main() {
```

```
    int n, cateto1, cateto2, hipotenusa;
```

```
    int achou;
```

```
    printf("Digite o comprimento maximo da hipotenusa: ");
```

```
    scanf("%d", &n);
```

```
    for (hipotenusa = 1; hipotenusa <= n; hipotenusa++)
```

```
    {
```

```
        achou = 0;
```

```
        /* testa todos os candidatos a cateto */
```

```
        for (cateto1 = 1; cateto1 < hipotenusa && !achou; cateto1++)
```

```
        {
```

```
            cateto2 = cateto1; /* cateto1 e' o menor cateto */
```

```
            while (cateto1*cateto1 + cateto2*cateto2 < hipotenusa*hipotenusa)
```

```
                cateto2++;
```

```
            if (cateto1*cateto1 + cateto2*cateto2 == hipotenusa*hipotenusa)
```

```
            {
```

```
                printf("hipotenusa = %d, catetos %d e %d\n", hipotenusa, cateto1, cateto2);
```

```
                achou = 1;
```

```
            }
```

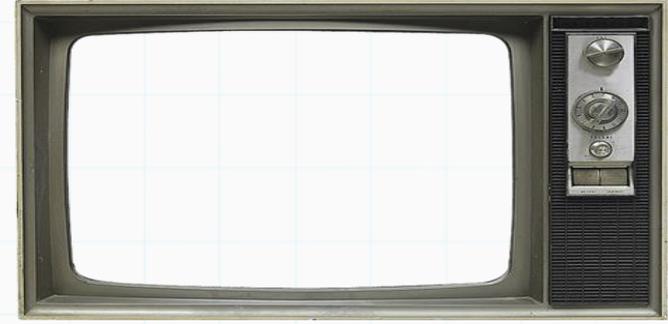
```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

Até a próxima



Slides baseados no curso de Aline Nascimento