

INSTITUTO FEDERAL

Norte de Minas Gerais

Campus Januária

Estruturas de Dados I

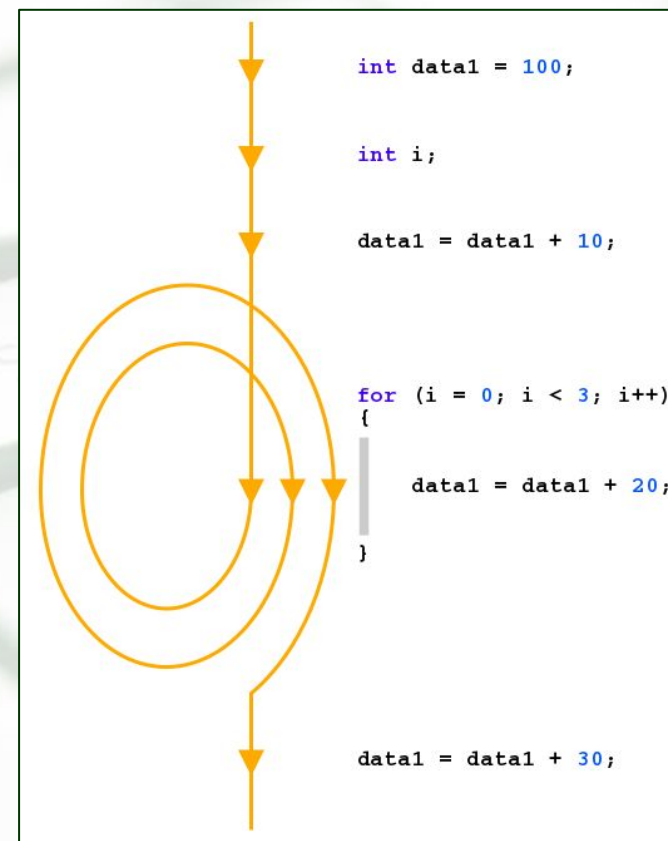
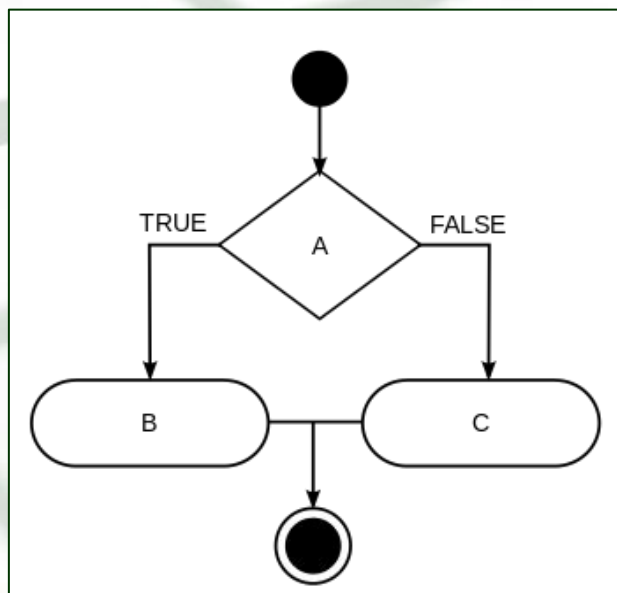
- *Controle de Fluxo* -



Controle de Fluxo

■ Estruturas para Controle de Fluxo de Execução

- *Estruturas Condicionais*
- *Estruturas de Repetição*





Estruturas Condicionais

■ Estruturas Condicionais

▫ **if-else**

```
if (condicao){  
    instrucao_1;  
    instrucao_2;  
    ...  
    instrucao_n;  
}
```

```
if (condicao) {  
    instrucao_1;  
    instrucao_2;  
    instrucao_n;  
} else {  
    instrucao_x;  
    instrucao_y;  
}
```



Operadores Lógico-Relacionais

>	Maior que
>=	Maior ou Igual
<	Menor que
<=	Menor ou Igual
=	Atribuição de Valor
==	Igualdade
!=	Diferente
&&	And (E)
	Or (Ou)
!	Not (Não)
0	False
1	True (<i>Qualquer valor !=0 é considerado Verdadeiro</i>).



Estruturas Condicionais

- Estruturas Condicionais
 - **if-else**

```
if (x%2)
    printf("Valor X é ____\n");
else
    printf("Valor X é ____\n");
```

Onde coloco PAR
Onde coloco ÍMPAR

?



Estruturas Condicionais

- Estruturas Condicionais
 - **if-else**

```
if (x%2)
    printf("Valor X é ÍMPAR\n");
else
    printf("Valor X é PAR\n");
```

Em C (e praticamente todas linguagens) qualquer valor diferente de 0 é considerado *TRUE*.



Estruturas Condicionais

- Estruturas Condicionais
 - **if-else**

Quando a execução **for de uma única instrução** não há necessidade de definir início e fim de bloco com **{ }**

```
if (x%2)
    printf("Valor X é ÍMPAR\n");
else
    printf("Valor X é PAR\n");
```

Em C (e praticamente todas linguagens) qualquer valor diferente de 0 é considerado *TRUE*.



Estruturas Condicionais

- Estruturas Condicionais
 - **if-else**

MAS CUIDADO COM
SENTENÇAS AMBÍGUAS...

```
int laranjas = 2;  
if (laranjas) //verifica se existe laranjas  
    if (laranjas > 3)  
        printf("Você consegue fazer um suco.\n");  
else  
    printf("Você não possui laranjas.\n");
```

O que será impresso?



Estruturas Condicionais

- Estruturas Condicionais
 - **if-else**

MAS CUIDADO COM
SENTENÇAS AMBÍGUAS...

```
int laranjas = 2;  
if (laranjas){ //verifica se existe laranjas  
    if (laranjas > 3)  
        printf("Você consegue fazer um suco.\n");  
}else  
    printf("Você não possui laranjas.\n");
```

O que será impresso?



Estruturas Condicionais

■ Operador Ternário

- *Forma simplificada do if-else*

```
condicao? expressaoTrue : expressaoFalse;
```

- *Exemplo*

```
int maior = x >= y? x : y;
```



Estruturas Condicionais

***Preze pela organização
do seu código...***

- Evite grandes conjuntos de **if-else** aninhados.
- Isso dificulta o entendimento do código, a manutenção, e correção de falhas.

```
if(test == a){  
    ...  
    ...  
}else  
    if(test == b){  
        ...  
        ...  
    }else  
        if(test == c){  
            ...  
            ...  
        }else  
            if(test == d){  
                ...  
                ...  
            }  
}
```



Estruturas Condicionais

■ *Switch-Case*

- Opção viável para evitar vários if-else aninhados
- Não é necessário delimitar cada bloco com chaves { }
- Comportamento em “cascata” se não for usado a instrução break

```
int opcao;  
scanf(" %d", &opcao);  
  
switch (opcao){  
    case 1: bloco1;  
            break;  
    case 2: bloco2;  
            break;  
    case 3: bloco3;  
            break;  
    case 4: bloco4;  
            break;  
    default: blocoDefault;  
}
```



Estruturas Condicionais

■ *Switch-Case* com avaliação em intervalos.

```
int temperatura;  
scanf(" %d", &temperatura);  
  
switch (temperatura){  
    case 0 ... 16: bloco1;  
                    break;  
    case 17 ... 26: bloco2;  
                    break;  
    case 26 ... 35: bloco3;  
                    break;  
    default: blocoDefault;  
}
```




Estruturas Condicionais

■ *Switch-Case* com avaliação múltipla.

```
int opcao;  
scanf(" %c", &opcao);  
  
switch (opcao){  
    case 'a':  
    case 'A': bloco1;  
               break;  
    case 'd':  
    case 'D': bloco2;  
               break;  
    //...  
}
```



Bora CODAR!!!



1. Faça um programa que leia dois números inteiros e imprima se eles são múltiplos ou não.
2. Programe uma calculadora IMC. Leia as informações de altura e peso e informe ao usuário se ele está abaixo do peso, com peso normal, acima do peso ou obeso.
3. Leia três notas de um aluno e calcule a média ponderada, considerando que o peso para a maior nota seja 4 e para as duas restantes, 3. Imprima uma mensagem “APROVADO” se a média for maior ou igual a 6 ou “REPROVADO” caso contrário.
4. Faça um programa que leia 3 valores (inteiros e positivos) de retas e verifique se eles conseguem formar ou não um triângulo.



Bora CODAR!!!



5. Faça um programa que leia as 5 notas de um quesito da Escola de Samba, descarte a maior e a menor nota, e apure a média das notas restantes.
6. Utilizando a estrutura *Switch-Case* faça um programa que leia do usuário um valor inteiro e imprima o nome do mês correspondente (ou se o mês não existe).
7. *A Fórmula de Bhaskara é uma das mais importantes da matemática, pois é utilizada para resolução das equações de segundo grau. Faça um programa que leia os valores A, B e C e calcule o resultado. (Ex.: Para A=1, B=-5, C=6; Resultado >> X1==3; X2==2)*
8. Problemas simples do cotidiano podem representar desafios para o mundo computacional. Faça um programa que, dados três números inteiros representando dia, mês e ano, imprima qual será o dia seguinte.



Laços de Repetição

- O comando **for** executa um número **determinado de repetições**, utilizando um contador de iterações.

```
for (inicializacao; condicao; incremento){  
    (...);  
    //instruções  
    (...);  
};
```



Laços de Repetição

■ Exemplo

```
for(int i=0; i<100; i++){  
    printf("Number: %d\n",i);  
};
```

BOA PRÁTICA

A declaração da variável de controle pode ser feita na própria estrutura.



Laços de Repetição

■ Exemplo

```
for(int i=0; i<100; i++){  
    printf("Number: %d\n",i);  
};
```

BOA PRÁTICA

*Salvo em casos específicos,
iniciá-la com valor 0*



Laços de Repetição

- O comando **while (enquanto)** avalia uma condição antes de iniciar as iterações.

```
while(condicao){  
    (...);  
    //instruções  
    (...);  
};
```



Laços de Repetição

■ *Exemplo*

```
int i=0;  
while(i<100){  
    printf("Number %d\n",i);  
    i++;  
};
```



Laços de Repetição

- O comando **do (repita)** executa o bloco de instruções pelo menos uma vez, testando a condição de parada somente ao final.

```
do{  
    (...);  
    //instruções  
    (...);  
}while(condicao);
```



Laços de Repetição

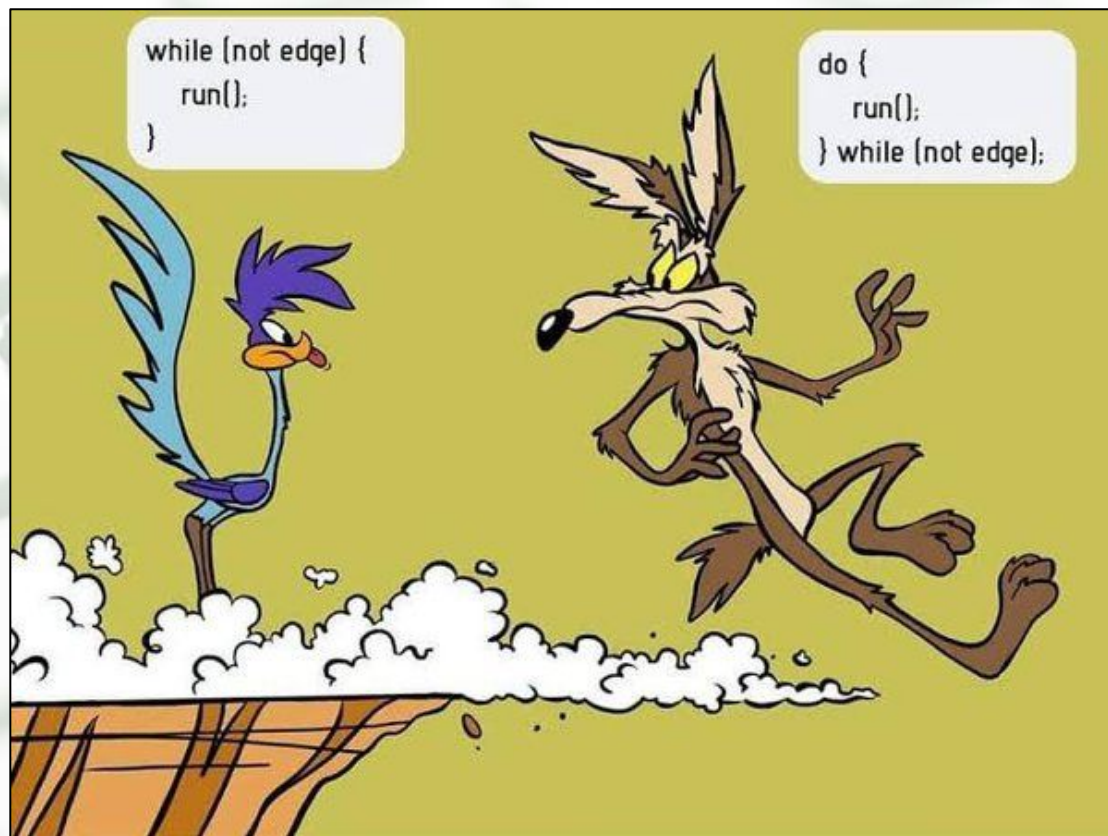
■ *Exemplo*

```
int i=0;  
do{  
    printf("Number %d\n",i);  
    i++;  
}while(i<100);
```




Laços de Repetição

- **while()** ou **do-while()** ?





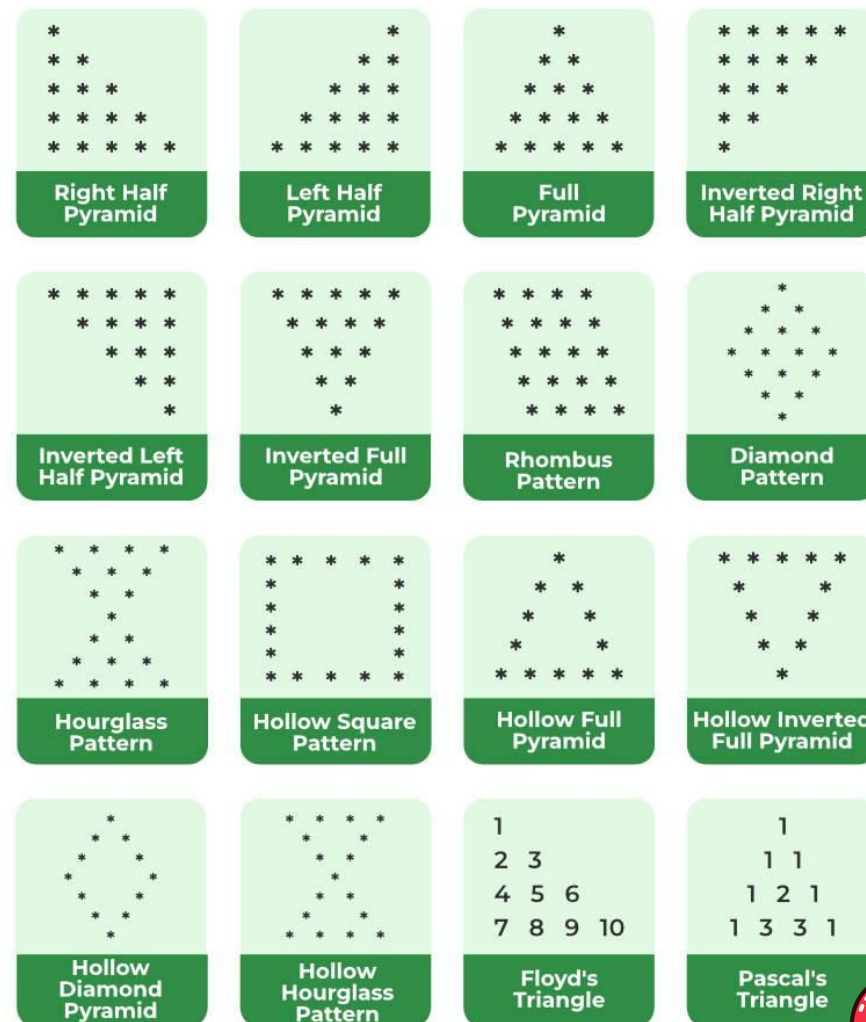
INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora CODAR!!!



Um excelente exercício para
treinar e praticar laços de
repetição e estruturas
condicionais são os
**problemas para
impressão de desenhos,
ou *print patterns***

Se ainda tem dificuldades
nestes conceitos,
NÃO DEIXE DE EXERCITAR...





Interrupção de Laços

break;

- A instrução **break** interrompe qualquer **laço de repetição** (for/do/while) e também finaliza uma entrada da estrutura **switch-case**;

```
do{  
    char opt = getchar();  
    if (opt == 'x')  
        break;  
    ...  
}while(1);
```

*Técnica chamada de
early-return*



Desvio

`continue;`

- A instrução **`continue`** ignora a iteração corrente de qualquer laço de repetição (`for`/`do`/`while`), reiniciando o laço na próxima iteração.

```
for (int x=0; x<100; x++){  
    if (x % 3)  
        continue;  
    printf("%d\n",x);  
}
```

O que será impresso pelo código ao lado?



Funções Úteis

```
system("clear");
```

- Biblioteca <stdlib.h>
- Comando para limpar o terminal de impressão...

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

int main(){
    printf("olá mundo.\n");
    system("clear");
    printf("OLÁ MUNDO.\n");
    return 0;
}
```




Funções Úteis

```
system("clear");
```

- Biblioteca <stdlib.h>
- Comando para limpar o terminal de impressão...

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

int main(){
    printf("olá mundo.\n");
    system("clear");
    printf("OLÁ MUNDO.\n");
    return 0;
}
```

*Teste com e
sem esse \n*



Função de Sorteio Aleatório

```
int rand(void);
```

- Biblioteca <stdlib.h>
- Sorteia um número inteiro “aleatório”.

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

int main(){
    int n;
    n = rand();
    printf("%d\n",n);
    return 0;
}
```



Função de Sorteio Aleatório

```
int rand(void);
```

- Biblioteca <stdlib.h>
- Sorteia um número inteiro “aleatório”.

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

int main(){
    int n;
    n = rand();
    printf("%d\n",n);
    return 0;
}
```

1º Problema

*O número sorteado
é muito grande!*

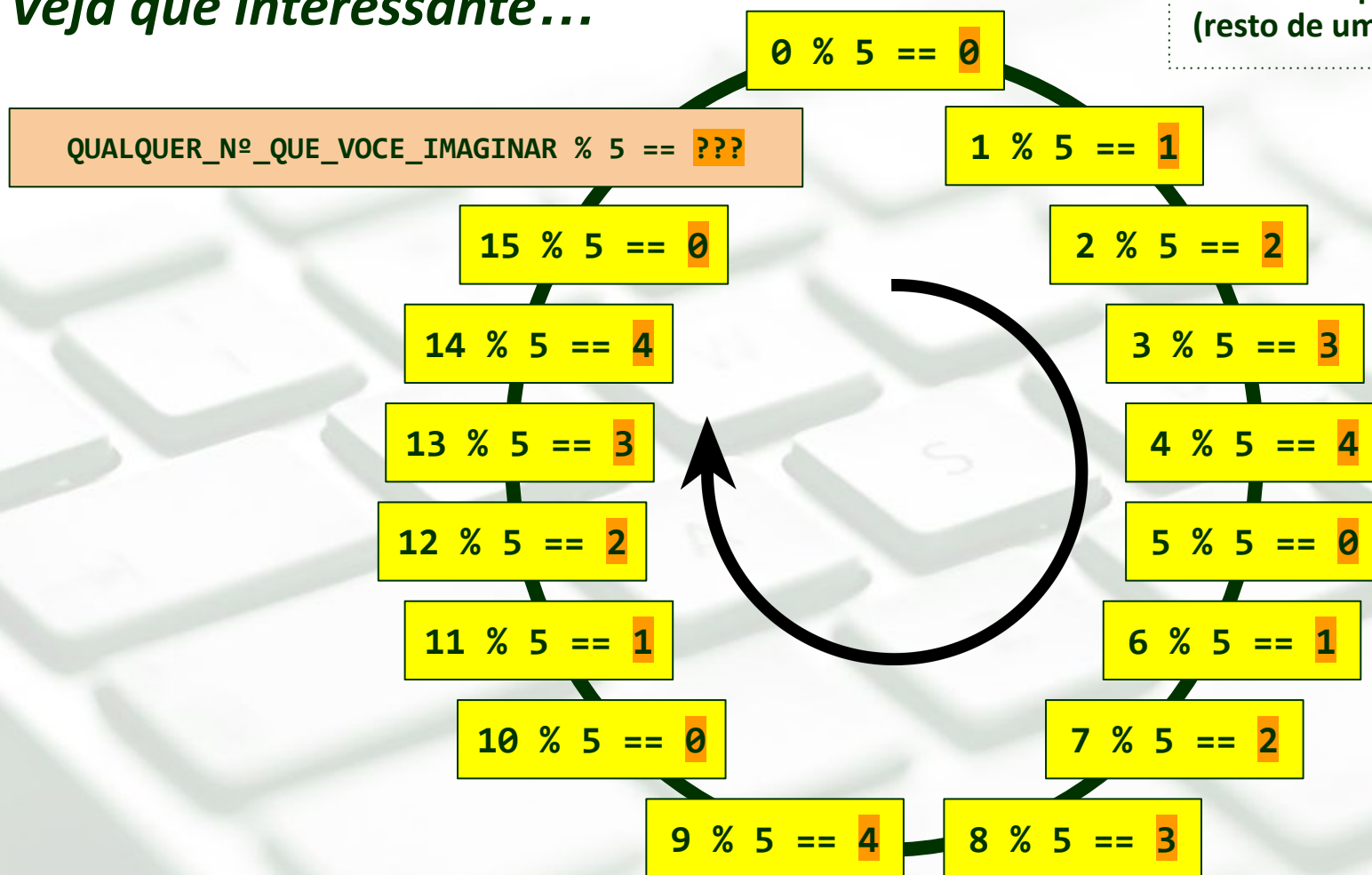
*Deseja-se sortear um
número aleatório,
mas em um intervalo
limitado.*



Função de Sorteio Aleatório

Veja que interessante...

% => Operador MOD
(resto de uma divisão inteira)



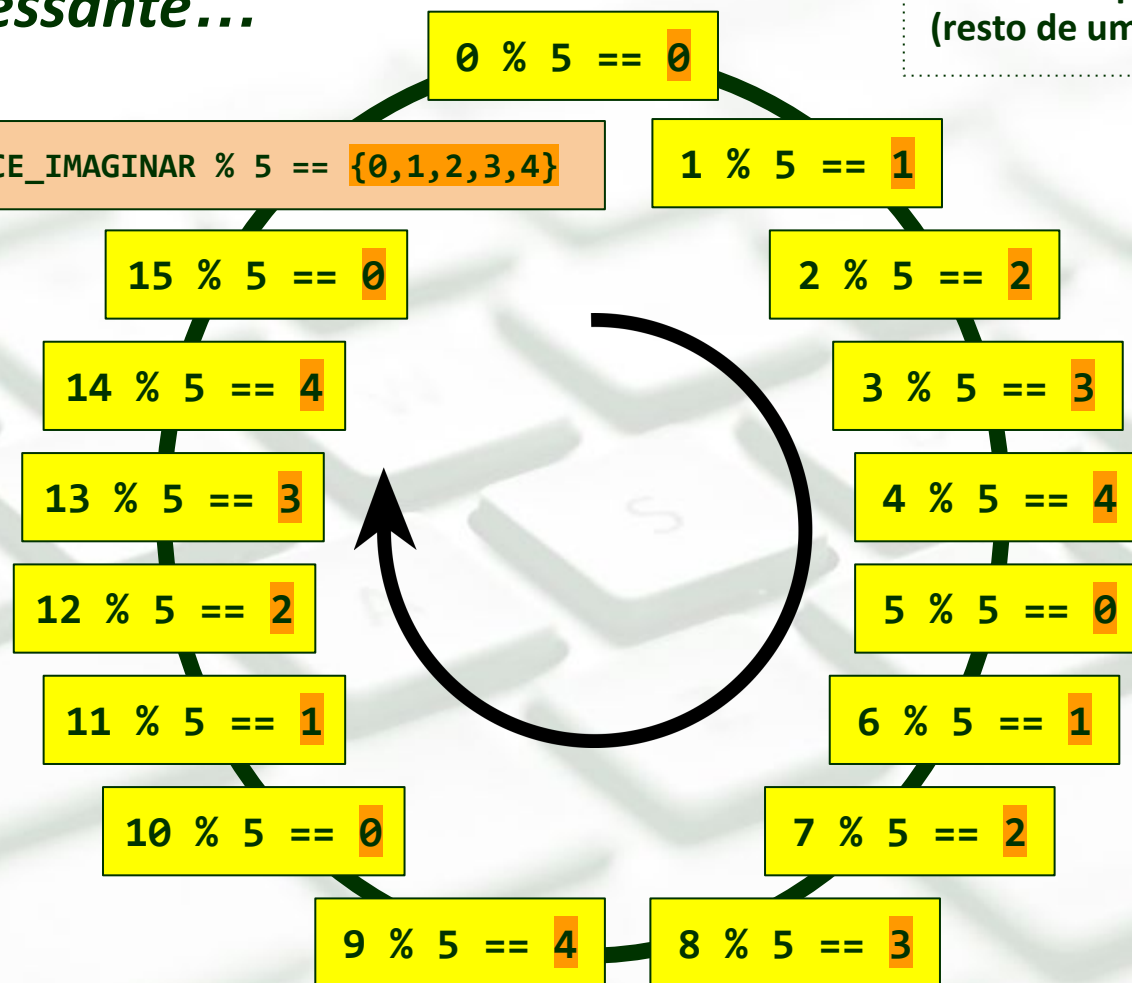


Função de Sorteio Aleatório

Veja que interessante...

% => Operador MOD
(resto de uma divisão inteira)

QUALQUER_Nº_QUE_VOCE_IMAGINAR % 5 == {0,1,2,3,4}

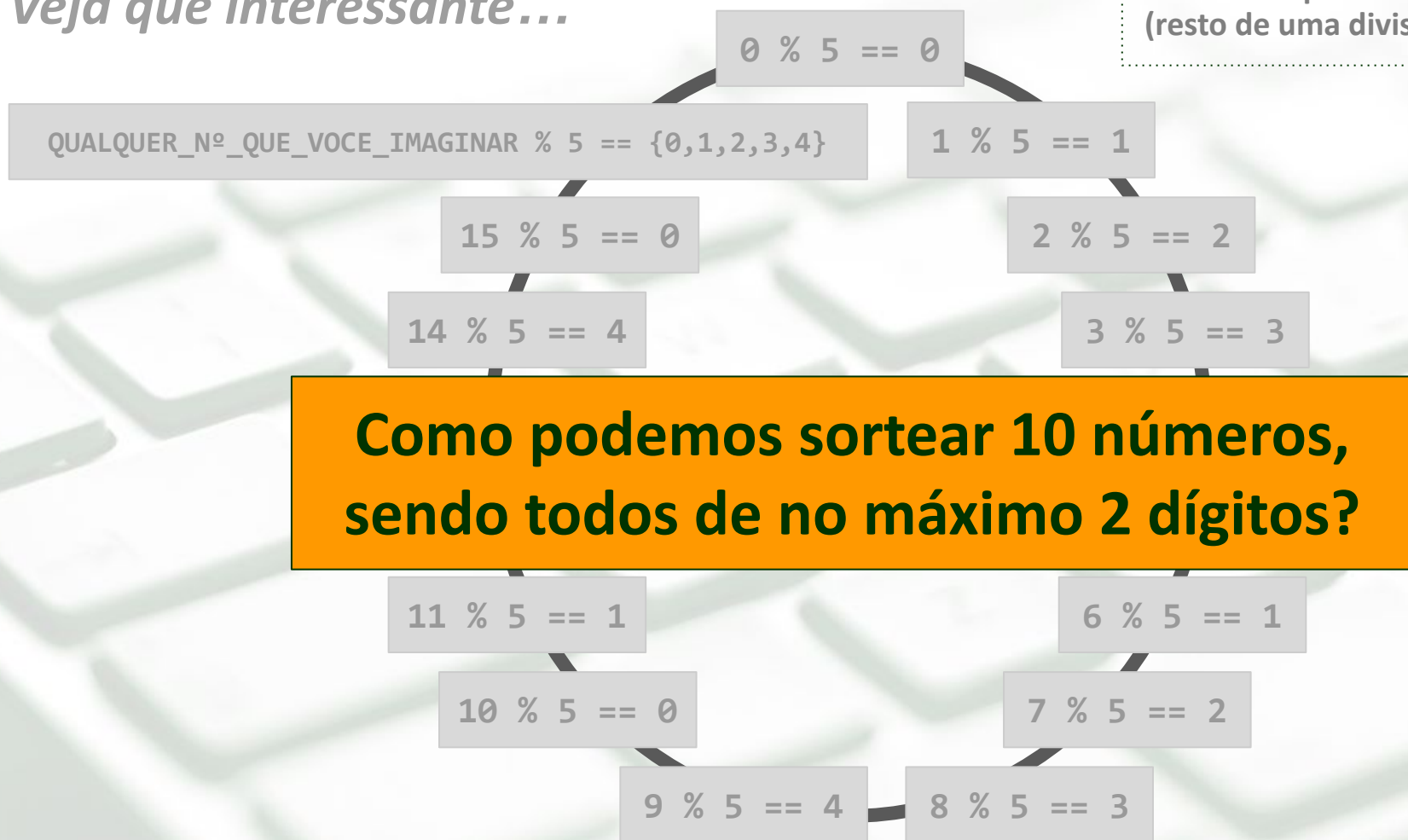




Função de Sorteio Aleatório

Veja que interessante...

% => Operador MOD
(resto de uma divisão inteira)





Função de Sorteio Aleatório

```
rand() % 100;
```

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

int main(){
    for(int i=0; i<10; i++){
        int n = rand() % 100;
        printf("%d\n", n);
    }
}
```



Função de Sorteio Aleatório

```
rand() % 100;
```

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

int main(){
    for(int i=0; i<10; i++){
        int n = rand() % 100;
        printf("%d\n", n);
    }
}
```

2º Problema

Os números sorteados são sempre os mesmos em cada execução.

Deseja-se sortear números aleatórios diferentes em cada execução.



Função de Sorteio Aleatório

```
rand() % 100;
```

Todo computador é **determinístico**, ou seja, completamente incapaz de gerar números aleatórios.
(para isso seria necessário ter imaginação)

Computadores geram na verdade **NÚMEROS PSEUDO-ALEATÓRIOS** que são números resultantes de cálculos matemáticos, iniciados por uma semente ou *seed* (valor inicial gerador).

2º Problema

Os números sorteados são sempre os mesmos em cada execução.

Deseja-se sortear números aleatórios diferentes em cada execução.



Função de Sorteio Aleatório

```
rand() % 100;
```

Todo computador é **determinístico**, ou seja, completamente incapaz de gerar números aleatórios.
(para isso seria necessário ter imaginação)

Computadores geram na verdade **NÚMEROS PSEUDO-ALEATÓRIOS** que são números resultantes de cálculos matemáticos, iniciados por uma semente ou *seed* (valor inicial gerador).

Portanto, quando alteramos a semente, todos os valores gerados pela função randômica serão distintos.

2º Problema

Os números sorteados são sempre os mesmos em cada execução.

Deseja-se sortear números aleatórios diferentes em cada execução.



Função de Sorteio Aleatório

```
int srand(unsigned int seed);
```

- Biblioteca <stdlib.h> para srand()
- Biblioteca <time.h> para time(NULL)
- **Altera a semente de geração de números aleatórios.**

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

int main(){
    srand(time(NULL));
    for(int i=0; i<10; i++){
        int n = rand() % 100;
        printf("%d\n", n);
    }
}
```

Essa instrução deve ser executada apenas uma vez durante a execução do programa, independentemente da quantidade de n^{os} que forem gerados.

Bora CODAR!!!



1. Faça um programa em C que leia do usuário dois números inteiros X e Y. Após isto, o programa deve imprimir X números aleatórios sorteados entre 0 e Y.
2. Desenvolva um programa que gere e imprima vários números aleatórios entre -100 e 100 (inclusive). O programa deve encerrar quando sortear o número zero. Ao fim imprima a quantidade de números obtidos.
3. O número 5.832 possui a seguinte característica: $5+8+3+2 == 18$ e $18^3 == 5.832$. Faça um programa que verifique se existe(m) outro(s) número(s) de quatro algarismos que possuem essa mesma característica.
4. Leia um número inteiro X. O programa deve gerar e imprimir vários números aleatórios sorteados no intervalo $[0, X]$. O programa só deve encerrar a execução quando sortear e imprimir um número primo.
5. Faça um programa que imprima o calendário de um mês (no formato de quadro). O usuário deve informar quantos dias tem no mês e o dia da semana em que se inicia (considere 1==domingo; 2==segunda; 3==terça; etc...).



Conteúdo Extra

- Biblioteca Externa GConio -



Instalação

- A **gconio.h** é uma *Lib* simplificada contendo diversas funções para controlar aspectos visuais de entrada/saída do terminal console.
 - **Gnu CONsole In-Out (gconio.h)**
- Para instalar a biblioteca, abra o terminal no mesmo diretório que contém o arquivo **gconio.h** e execute:

```
# sudo cp gconio.h /usr/include
```

- Reinicie o Geany.



Principais Funções

```
int textcolor(COR_DA_LETRA);
```

```
int textbackground(COR_DO_FUNDO);
```

- Altera cores padrão utilizadas na saída do programa.
- Cores definidas pelas constantes:
 - BLACK, RED, GREEN, BROWN, BLUE, PURPLE, CYAN, WHITE.

```
int resetcolor()
```

- Restaura a configuração padrão de cores do terminal.



Principais Funções

```
void clrscr()
```

- Limpa a área de impressão da tela.

```
void clreol();
```

- Limpa a linha atual do cursor.

```
int get_screen_columns();
```

```
int get_screen_rows();
```

- Obtém a quantidade de colunas e linhas do terminal.



Principais Funções

```
void gotoxy(int X, int Y)
```

- Posiciona o cursor de impressão da tela nas coordenadas X e Y. Em modo padrão, o terminal de impressão possui 24 linhas x 80 colunas.

```
void cursor(int enabled)
```

- Habilita/Desabilita o cursor na tela do terminal.

```
void delay(int T);
```

- Interrompe a execução por T *milissegundos*
 - 1000 Milissegundos == 1 segundo.



Principais Funções

`int getch()`

- ❑ Realiza a leitura de uma tecla por vez.
- ❑ Não produz “eco” no terminal (leitura silenciosa).

`void kbhit(char* key)`

- ❑ Verifica (*em background*) se uma tecla foi acionada.
- ❑ A tecla acionada é registrada na variável `key`.

`void flushall()`

- ❑ Descarrega todos os buffers (`stdin` e `stdout`).



Bora Codar (1)



- Faça um programa que simule um semáforo...
 - 10 segundos tela VERDE
 - 3 segundo tela AMARELA
 - 10 segundos tela VERMELHA
- Os três últimos segundos da tela verde e vermelha devem ser contados regressivamente e mostrados na tela.
- Use apenas as funções da lib `gconio.h`



Bora Codar (2)



- Faça um programa que leia as coordenadas de 2 pontos de um retângulo (superior esquerdo e inferior direito), valide os valores e renderize o retângulo na tela.

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
Informe Coordenadas (X,Y) de P1: 5 5
Informe Coordenadas (X,Y) de P2: 75 15

[Green Rectangle]

-----
(program exited with code: 0)
Press return to continue
```




Bora Codar (3)



- Faça um programa que produza a seguinte saída:

```
Terminal
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
Tamanho de cada casa (L x A): 5 2
[Grid of 10x8 alternating green and gray squares]
```