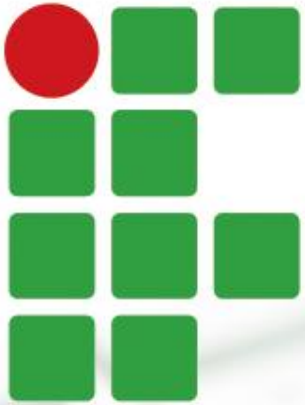


INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados I

- Linguagem C -

Breve Histórico...

- Criada por **Dennis Ritchie** em 1972, nos laboratórios da *AT&T*, para implementação do sistema **UNIX**.
- UNIX permitiu uma grande difusão da linguagem.
- Uma das linguagens mais adotadas no mundo, e compatível com praticamente todas as arquiteturas existentes.
- Originou o C++ (suporte à orientação a objetos) e influenciou diversas outras linguagens
  - PHP, C#, Java, Shell, etc.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Ranking

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	**2**		**C**	**7.742%**	**-6.86%**
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%
5	5		Python	3.919%	-0.34%
6	7	▲	Visual Basic .NET	3.174%	+0.61%
7	6	▼	PHP	3.009%	+0.24%
8	8		JavaScript	2.667%	+0.33%
9	11	▲	Delphi/Object Pascal	2.544%	+0.54%
10	14	⇈	Swift	2.268%	+0.68%
11	9	▼	Perl	2.261%	+0.01%
12	10	▼	Ruby	2.254%	+0.02%
13	12	▼	Assembly language	2.232%	+0.39%
14	16	▲	R	2.016%	+0.73%
15	13	▼	Visual Basic	2.008%	+0.33%

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Ranking

Mar 2017	Mar 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.384%	-4.14%
2	**2**		**C**	**7.742%**	**-6.86%**
3	3		C++	5.184%	-1.54%
4	4		C#	4.409%	+0.14%

The C Programming Language

Some information about C:

Highest Position (since 2001): #1 in Mar 2015

Lowest Position (since 2001): #2 in Mar 2017

Language of the Year: 2008

13	12	⌄	Assembly language	2.232%	+0.39%
14	16	⌃	R	2.016%	+0.73%
15	13	⌄	Visual Basic	2.008%	+0.33%

Principais Características

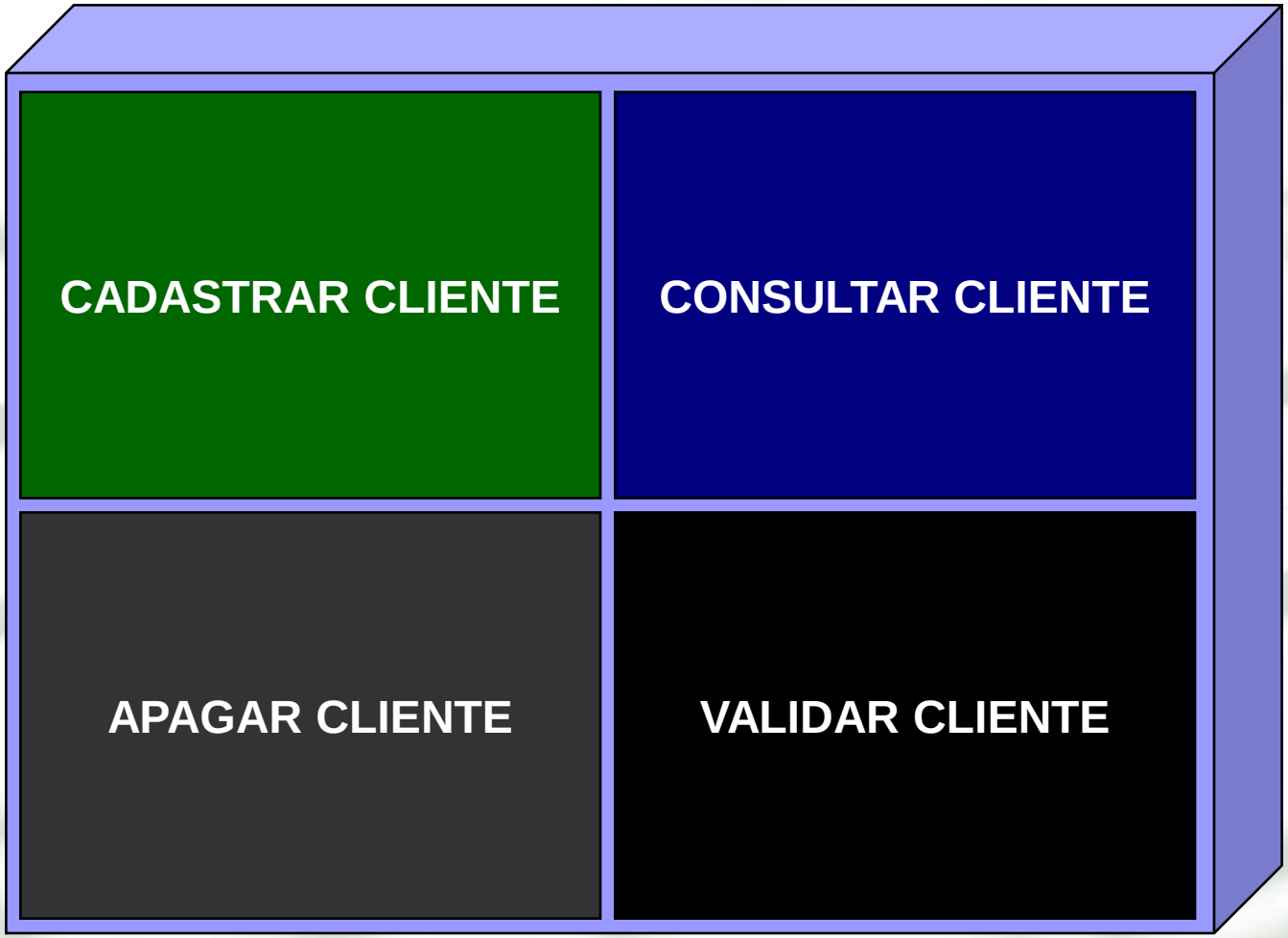
- Alto Nível (com muitos recursos de Baixo Nível)
  - Permite instruções em Assembly (*Machine Code*)
- Compilada
- Estruturada
- Estaticamente tipada
- Modular
- Case-sensitive
- Portável (Código Objeto / ANSI)
- Códigos mais enxutos => mais velozes
- Linguagem para profissionais
  - poucas restrições, poucas excessões, poucas palavras reservadas, etc...

Filosofia do C

- O que fazer para resolver um problema muito complexo?
- Dividi-lo em problemas menores?
- É mais fácil implementar pequenos pedaços de código que realizem corretamente uma única função (e a realizem bem) do que fazer extensos códigos, com muitas variáveis, condições e exceções de forma a atingir o mesmo resultado.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Filosofia do C



Ciclo de Desenvolvimento

- Editor
  - Código-Fonte
- Pré-Processador
  - Código Expandido
- Compilador
  - Código Objeto
- Linkeditor / Linker
  - Executável

Tipos Básicos da Linguagem

- **int** (valores numéricos inteiros) - 2 Bytes
  - **short** int (1B)
  - **long** int (4B)
  - **unsigned long** int (4B)
- **char**
  - Caracteres alfanuméricos - 1 Byte
- **float**
  - Valores com casas decimais - pouca precisão
- **double**
  - Grandes valores com casas decimais mais precisas - valores científicos - 8 bytes
- **void**
  - vazio / sem retorno

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Operadores Lógico-Relacionais

>	Maior que
>=	Maior ou Igual
=	Atribuição
==	Igual
!=	Diferente
&&	And (E)
||	Or (Ou)
!	Not (Não)
0	False
!=0	True

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Operadores Aritméticos

+	Soma
-	Subtração
/	Divisão
*	Multiplicação
%	Resto de Divisão - Módulo
+=	Atribuição Aritmética x += 1 :: x = x + 1
-=	Atribuição Aritmética x -= 2 :: x = x - 2
++	Incremento i++ :: i = i + 1
--	Decremento --i :: i = i - 1

Operadores Aritméticos

```
int a, b, c, i;
i = 3;      // a: ?   b: ?   c: ?   i: 3
a = i++;    // a: 3   b: ?   c: ?   i: 4
b = ++i;    // a: 3   b: 5   c: ?   i: 5
c = --i;    // a: 3   b: 5   c: 4   i: 4
```

Os operadores incrementais serão bastante utilizados em laços de repetição;

Estrutura Básica

Primeiro código em C...

```
#include <stdio.h>

int main(){
    printf("hello world!");
    return 0;
}
```

Regras de Sintaxe

- Todo programa em C consiste em uma ou mais **funções**.
- Todo programa sempre inicia a execução a partir da função **`main()`**
- Os caracteres **`{ }`** determinam o início e o fim de blocos de execução.
- //Linha de comentário
- /* Bloco de

  Comentários */

Declaração de Variáveis

```
#include <stdio.h>
int main(){
      float altura;
      int idade, soma;
      int qtdePessoas;
}
```

tipo nomeDaVariavel;

Declaração de Variáveis

```
#include <stdio.h>
int main(){
      float altura;
      int idade, soma;
      int qtdePessoas;
}
```

Pode-se declarar duas ou mais variáveis de um mesmo tipo em uma única instrução.

Declaração de Variáveis

```
#include <stdio.h>
int main(){
      float altura;
      int idade, soma=0;
      int qtdePessoas=0;
}
```

Também é possível inicializar a variável com um valor determinado.

Declaração de Variáveis

```
#include <stdio.h>
int main(){
      float altura;
      int idade, soma=0;
      int qtdePessoas=0;
}
```

BOA PRÁTICA!
O nome de uma variável deve ser escolhido de modo a facilitar a compreensão da sua utilidade/necessidade.

Declaração de Variáveis

```
#include <stdio.h>
int main(){
      float altura;
      int idade, soma=0;
      int qtdePessoas=0;
      float mediaAlturas;
      int opcaoUsuario;
}
```

BOA PRÁTICA!

Quando nomes compostos, a primeira inicial em minúsculo e as seguintes em maiúsculo:

Uso de Bibliotecas

- A linguagem C permite a importação de **bibliotecas**, possibilitando o uso de uma infinidade de funções pré-programadas.
- Por exemplo, as operações de **leitura em teclado** e **impressão em tela** são feitas através de funções da biblioteca **`stdio`**.
  - **`STanDart In/Out`**

```
#include <nome_da_biblioteca.h>
```

Estrutura Básica

```
#include <stdio.h>
int main(){

    printf("hello world!");
    return 0;
}
```

Indentação é Obrigatória!

ATENÇÃO

A **Indentação de Código**, além de ser uma boa prática, facilita a leitura, organização e correção dos algoritmos!

Você só tem a ganhar fazendo-a corretamente.

Exemplo

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

typedef struct{
int i;
num* prox; }num;

num* setNum(){
num* n = (num*)malloc(sizeof(num));
printf("Digite Num: ");
scanf(" %d",&n->i);
n->prox = NULL;
return n; }

num* getUltimoNum(num* list){
if(list->prox)
return getUltimoNum(list->prox);
else
return list; }
```

```
void getNums(num* list){
if(list){
printf("\n%d",list->i);
getNums(list->prox); }  }

int main(){
num* fila = NULL;
int opt;
do{
if(!fila)
fila = setNum();
else{
num* fim = getUltimoNum(fila);
fim->prox = setNum(); }

printf("Continua?");
scanf(" %d",&opt);
}while(opt);
getNums(fila);
getch();
}
```

Exemplo

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

typedef struct{
    int i;
    num* prox;
}num;

num* setNum(){
    num* n = (num*)malloc(sizeof(num));
    printf("Digite Num: ");
    scanf(" %d",&n->i);
    n->prox = NULL;
    return n;
}

num* getUltimoNum(num* list){
    if(list->prox)
       return getUltimoNum(list->prox);
    else
       return list;
}
```

```c
void getNums(num* list){
   if(list){
      printf("\n%d",list->i);
      getNums(list->prox);
   }
}

int main(){
    num* fila = NULL;
    int opt;
    do{
       if(!fila)
         fila = setNum();
       else{
         num* fim = getUltimoNum(fila);
         fim->prox = setNum();
       }
       printf("Continua?");
       scanf(" %d",&opt);
    }while(opt);
    getNums(fila);
    getch();
}
```

Regras de Sintaxe

- Em C, **tudo** é armazenado “como número”, inclusive letras e caracteres especiais.

```
int main(){
    char a,b,c;
    a = 65;
    b = ‘B’;
    c = a+b;
}
```

Exemplo

```
int main() {
    int x = 3;
    float f = 1.5;
    char ch = ‘a’;
    ch = x;
    x = f;
    f = ch;
    x = f = ch = 0;
    return 0;
}
```

Saída Formatada: printf()

```
int printf(const char* st_contr [, lista_args]);
```

- Biblioteca **<stdio.h>**

- A função **printf** é o comando padrão do C para impressão no dispositivo de saída *default* (monitor);
- A string de controle (*st_contr*) é a máscara que especifica o que será impresso e de que maneira será impresso (inclusive as variáveis presentes na *lista_args).*

Exemplos

- `printf("Olá Mundo");`
- `printf("Linha 1 \nLinha 2");`
- `printf("Coluna1 \tColuna 2");`
- `print("Beep \a");`

- `printf("Tenho %d Anos de Vida",26);`
- `printf("Total da Conta: %f",total);`
- `printf("Media %.2f", soma/qtde);`

Caracteres de Escape

\n	Nova Linha
\a	Campainha
\t	Tabulação Horizontal
\v	Tabulação Vertical
\’	Apostrofo
\”	Aspas
\\	Barra Invertida
\0	Caracter Nulo

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Exercício A

- Desenvolva dois programas que produzam as seguintes saídas:
  - *Obs: Utilize **um único** comando **`printf()`** e **espaçamento simples** entre as palavras.*

```
Bacharelado    em

        Sistemas       de      Informação
```

```
        \\Meu Nome é: “Adriano”
```

Caracteres de Formatação

%c	**C**aracter Simples
%d - %i	**I**nteiro **D**ecimal
%2d - %**0**2i	**I**nteiro com duas casas
%e	Notação Científica
%f	Ponto **F**lutuante
%3.2f	Ponto **F**lutuante com Arredondamento
%o	**O**ctal
%s	**S**tring
%u	Decimal sem Sinal (**U**nsigned)
%x	He**x**adecimal

Exercício B

- A partir do código abaixo, faça um programa que produza a saída ao lado:
  - OBS: utilize um único comando **printf().**

```
int main() {
    float j = 23.692;
    int k = 65;
}
```

```
23.7
0065
A
23.69
```

Typecast

- Qual será a saída gerada pelo código abaixo?

```
int main() {
    int x,y;
    float f;
    x = 3;
    y = 2;
    f = x/y;
    printf("%2.2f", f);
    return 0;
}
```

Typecast

! Atenção !

- Operações aritméticas realizadas entre variáveis de mesmo tipo resultam em um valor do mesmo tipo.
- Exemplo:

```
int/int == int
```

Mesmo que o resultado não seja inteiro.

Typecast

- Para resolver essas situações, utiliza-se a conversão explícita de tipos (***TYPECAST***).
- Exemplo:

```
(float)int/int == float
```

- A instrução **`(float)`** antes da operação aritmética indica para o compilador que espera-se um resultado do tipo **`float`**.

Typecast

- Tente novamente...

```
int main() {
    int x,y;
    float f;
    x = 3;
    y = 2;
    f = (float)x/y;
    printf("%2.2f", f);
    return 0;
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Leitura Formatada – scanf()

```
int scanf(const char* st_contr [, end_var, ...]);
```

- Biblioteca `<stdio.h>`

- A função **scanf** é o comando padrão do C para leitura no dispositivo de entrada *default* (teclado);
- A lista de argumentos (*end_var*) deve consistir nos **endereços** das variáveis, obtido através do operador de endereço **&**;

Exemplos

- `scanf("%d",&num);` /* lê num como **int** */
- `scanf("%f",&num);` /* lê num como **float** */
- `scanf("%c",&num);` /* lê num como **char** */

Teste:

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

Sugestão: Utilize o `scanf` para apenas uma leitura por vez.

Atenção!

- Analise o código abaixo:

```
int main () {
      char j,k;
      scanf("%c",&j);
      scanf("%c",&k);
      printf("%c\n%c",j,k);
      return 0;
}
```

- O será impresso na tela?

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Solução para o Problema

DICA: Espaço em branco sempre no início do scanf()

```
int main () {
      char j,k;
      scanf(" %c",&j);
      scanf(" %c",&k);
      printf("%c\n%c",j,k);
      return 0;
}
```

Exercícios C

1. Faça um programa em C que leia do usuário dois valores reais e retorne o produto destes números com duas casas decimais.

2. Faça um programa que leia do usuário um símbolo (caractere) e retorne o código ASCII correspondente.

3. Faça um programa que recebe um caractere minúsculo (a-z) e imprima o mesmo caractere em maiúsculo.

4. Programe uma bomba de combustível: o usuário informa o preço do litro de combustível e o valor que o motorista deseja abastecer. Informe a quantidade de combustível que a bomba irá dispensar.

5. Joãozinho tem um cofre com muitas moedas, e deseja saber quantos reais conseguiu poupar. Faça um programa para ler a quantidade de cada tipo de moeda, e imprimir o valor total economizado, em reais.

Exercícios C

6. Programe um caixa eletrônico. O usuário deve informar o valor que deseja sacar e o programa emite a menor quantidade de notas possíveis, totalizando o valor (Notas disponíveis: 100, 50, 20, 10, 5, 2 e 1).

7. Faça um programa em C que leia três números inteiros e imprima a média simples (a média pode ser um número real).

8. Faça um programa que leia dois símbolos numéricos (0 a 9) no formato char e imprima o resultado do produto dessa multiplicação.

9. Faça um programa que leia o tempo da duração de um evento expressa em segundos e mostre-o expresso em horas, minutos e segundos.

10. O custo final de um produto qualquer é a soma do custo de produção, acrescido de X% do distribuidor e Y% de impostos. Faça um programa que dadas os parâmetros informe o custo final de um produto.

Estruturas de Controle

- Estruturas Condicionais

```
if (condicao){
      bloco;
}
```

```
if (condicao) {
      bloco1;
    } else {
         bloco2;
}
```

Estruturas de Controle

- Operador Ternário
  - *Forma simplificada do if-else*

```
condicao? expressao1:expressao2;
```

  - *Exemplo*

```
int valido = horas>24? 0:1;
```

Estruturas de Controle

- Estruturas Condicionais

```
int opcao;
scanf("%d", &opcao);

switch (opcao){
      case 1: bloco1;
      case 2: bloco2;
      case 3: bloco3;
      default: bloco4;
}
```

Exercícios D

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 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.

Exercícios D

5. Faça um programa 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 (para)** executa um número determinado de repetições, utilizando um contador de iterações.

```
for (inicializacao; condicao; incremento)
{
        (...);
        bloco;
        (...);
};
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Laços de Repetição

- *Exemplo*

```
for (int i=0; i<=100; i++) {

    printf("Escreva o Numero %d\n",i);

};
```

Laços de Repetição

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

```
while (condicao) {
        (...);
        bloco;
        (...);
};
```

Laços de Repetição

- *Exemplo*

```
while (i<=100) {

      printf("Escreva o Numero %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 {
      (...);
      bloco;
      (...);
} while (condicao);
```

Laços de Repetição

- *Exemplo*

```
do {

      printf("Escreva o Numero %d\n",i);
      i++;

} while (i<=100);
```

Interrupção de Laços

break;

- A instrução *break* serve para interromper um **laço de repetição** (for / do / while) ou terminar um conjunto switch-case;

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

Desvio

continue;

- A instrução *continue* serve para "saltar" uma iteração dentro de um laço de repetição (for / do / while), sem sair do laço.

```
for (x=0; x<100; x++) {
      if (x%2)
            continue;
      printf("%d\n",x);
}
```

Funções Úteis

system("clear")

- Biblioteca <stdlib.h>
- Comando para limpa a tela...

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

int main(){
      printf("olá mundo");
      system("clear");
      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 = rand();
      printf("%d", n);
      return 0;
}
```

Funções Úteis

int rand(void)

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

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

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

1º Problema

O número sorteado é muito grande!

Deseja-se sortear um número aleatório com valor entre 0 e 10

Função de Sorteio Aleatório

```
rand() % 11
```

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

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

int main(){
      int n = rand() % 11;
      printf("%d", n);
      return 0;
}
```

Funções Úteis

int rand(void)

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

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

int main(){
      int n = rand() % 11;
      printf("%d", n);
      return 0;
}
```

2º Problema

O número sorteado é sempre o mesmo!

Deseja-se sortear um número aleatório diferente a cada execução.

Função de Sorteio Aleatório

int srand(unsigned int seed)

- Biblioteca <stdlib.h>
- Altera a semente de geração de números aleatórios.

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

int main(){
        srand(time(NULL));
        int n = rand() % 11;
        printf("%d", n);
        return 0;
}
```

Exercícios E

1. Faça um programa em C que leia do usuário um número inteiro X. Após isto, o programa deve imprimir uma quantidade X de números aleatórios entre 0 e 50.

2. Desenvolva um programa que gere um número aleatório secreto entre 0 e 10 (não imprima esse número). Faça com que o usuário tente acertar o número sorteado. Quando acertar, informe quantas tentativas foram realizadas até o êxito.

3. Leia um número inteiro X. O programa deve gerar um número aleatório entre 0 e X. Após isto, o programa deve dizer se o número sorteado é primo ou não.

4. O número 3.025 possui a seguinte característica: 30 + 25 = 55 e 55 x 55 = 3.025. Escreva um programa que escreva todos os números com quatro algarismos que possuem essa mesma característica.

5. Faça um programa que imprima o calendário de um mês (no formato de quadro). O usuário deve informar quantos dias o mês e o dia da semana em que o mês se inicia.