

SI100

Algoritmos e Programação de Computadores I

1º Semestre - 2018

Tópico 2a:

Linguagem Estruturada e Introdução à Linguagem C

Unidades 3 e 4

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

Roteiro

- Algoritmos e Programas de Computador:
 - Revisão;
 - Linguagens de Programação;
 - Compiladores e Interpretadores;
- Introdução à Linguagem C:
 - Introdução;
 - Primeiro programa em C;
- Aspectos Fundamentais da Linguagem C:
 - Constantes e Variáveis;
- Exercícios;
- Referências.

ALGORITMOS E PROGRAMAS DE COMPUTADOR

Etapas para desenvolvimento de um programa

- O desenvolvimento de um programa de computador obrigatoriamente deve passar por *três etapas*:
 - **Análise:** nesta etapa estuda-se o *enunciado* do problema que se quer resolver para que sejam definidos:
 - Os dados de entrada;
 - O processamento que deverá ser feito;
 - Os dados de saída;
 - **Elaboração do algoritmo:** definição da sequência de passos que deverá ser realizada para que o objetivo seja atingido;
 - **Codificação:** consiste na *transformação* do algoritmo em códigos de uma *linguagem de programação* escolhida.

Algoritmos

- **Algoritmo** é um procedimento computacional que recebe valores de entrada e produz valores de saída;
 - “*Receita*” para solução do problema.
- Pode ser representado de diversas formas:
 - **Pseudo-código textual:**
 - Sujeito a ambiguidade!
 - **Representação gráfica** (em diagramas);
 - Mais **fiel à linha de raciocínio**;
 - Mais **clara** que a representação textual;
 - Exige o conhecimento das convenções de cada representação.
 - Ex.: fluxograma.

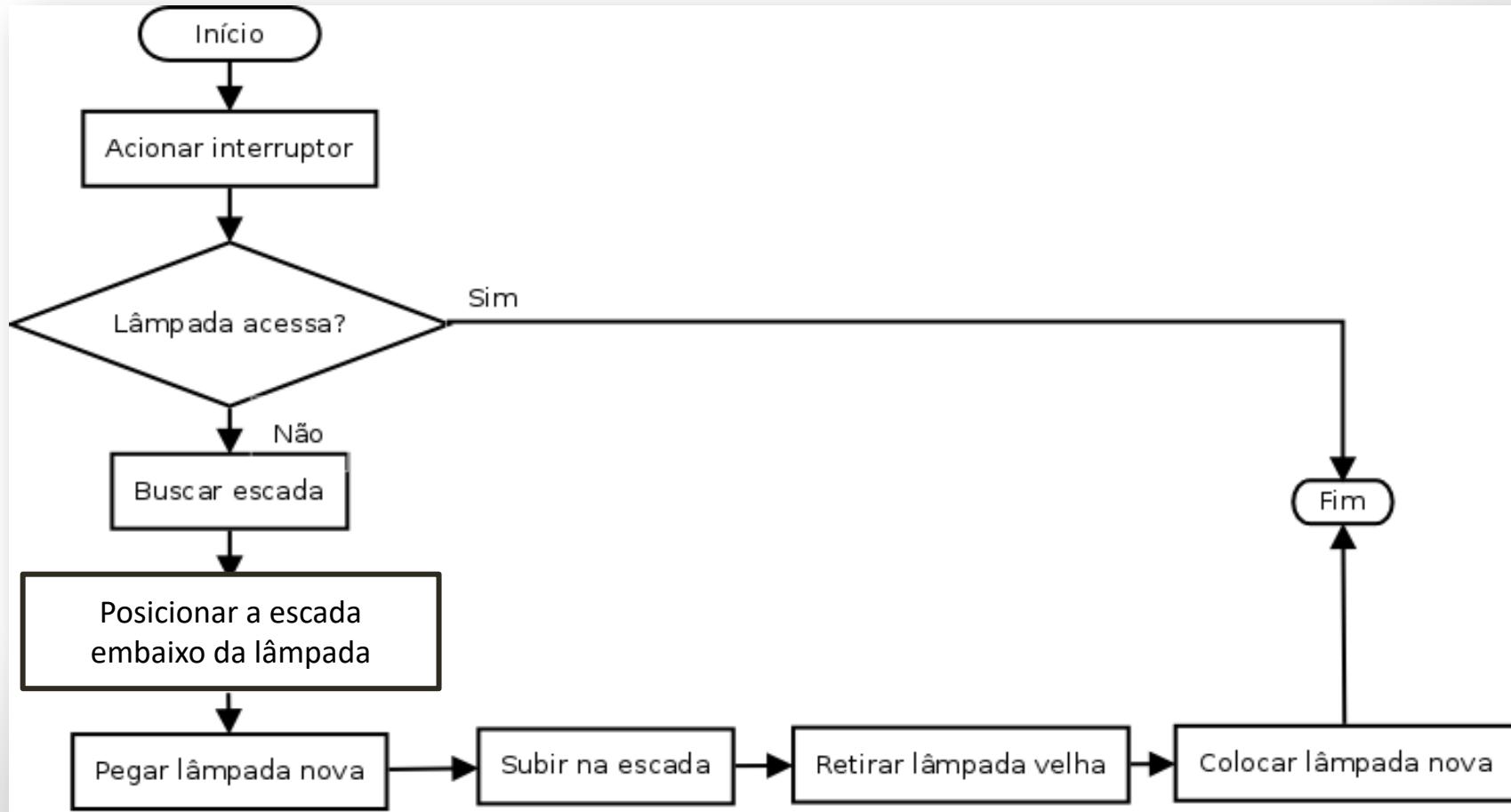
Algoritmos: pseudo-código

- Exemplo:

- 1) Acionar o interruptor;
- 2) Se a lâmpada **não acender**, então
 - 3) Buscar uma escada;
 - 4) Posicionar a escada embaixo da lâmpada;
 - 5) Pegar uma lâmpada nova;
 - 6) Subir na escada;
 - 7) Retirar a lâmpada velha;
 - 8) Colocar a lâmpada nova;



Algoritmos: fluxograma



Algoritmos → Programas

- Como transformar um algoritmo em um programa de computador?
 - É preciso usar uma linguagem que o computador entenda;
 - Tal linguagem deve ser capaz de expressar tudo que o computador pode fazer;
 - Não pode ser ambígua.



Linguagem de Programação

Programas

- **Linguagem de máquina:** conjunto de instruções que podem ser interpretadas e executadas diretamente pela CPU (códigos binários);
- **Linguagem de Programação:** sintaxe e semântica utilizada para escrever um programa
 - **Linguagem de baixo nível (*assembly*):** linguagem de programação baseada em mnemônicos, dependente do tipo de máquina e de fácil tradução para a linguagem de máquina.
 - **Linguagem de alto nível:** linguagem de programação que independe do conjunto básico de instruções da linguagem de máquina (C, Java, Python,...);

Programas

- Mas um computador só executa programas em linguagem de máquina;
- Como um programa em linguagem de alto nível (ou de montagem) poderia ser executado?

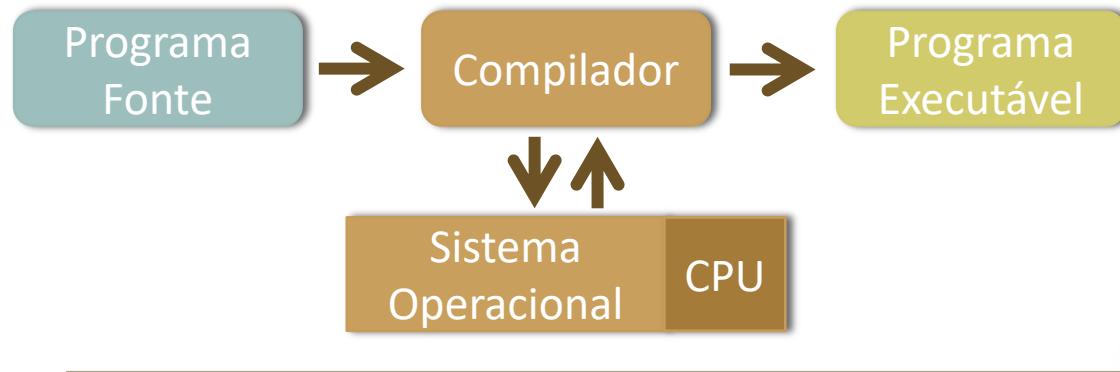
É preciso **convertê-lo!**

- Isto é feito por:
 - Compiladores; ou
 - Interpretadores.

Compiladores

- **Compilador:** programa que traduz os programas escritos em uma dada linguagem de programação para linguagem de máquina;

Geração do Programa Executável:



Execução do Programa:



Interpretadores

- **Interpretador:** programa que executa diretamente os programas escritos em uma dada linguagem de programação;



- Programas interpretados tendem a ser mais lentos;
- Por outro lado, podem ser **alterados durante a execução**.

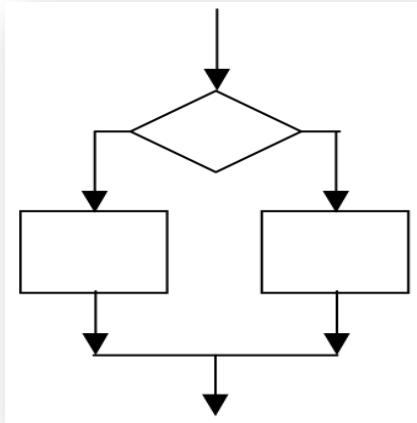
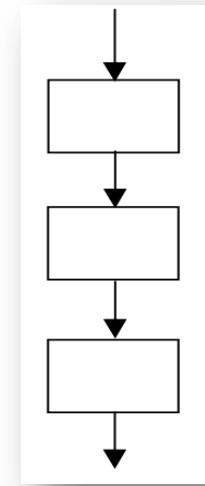
INTRODUÇÃO À LINGUAGEM C

A Linguagem de Programação C

- A linguagem **C** foi criada no *Bell Labs* em 1972;
- Rapidamente se tornou uma das linguagens de programação mais importantes e populares.
 - É a segunda linguagem mais utilizada pelo mercado - participação de 11,857% (Índice TIOBE, Fev. 2018 - <http://goo.gl/DMs2>);
- É uma linguagem de **programação estruturada**:
 - Programa composto por **blocos elementares** interligados por **três** mecanismos básicos:
 - Sequência;
 - Seleção;
 - Iteração.

A Linguagem de Programação C

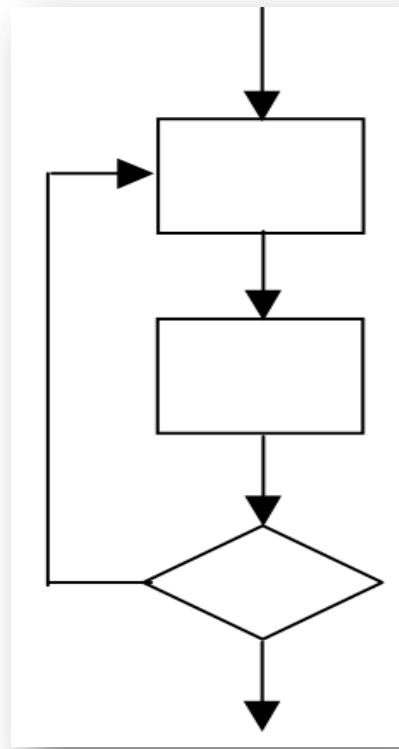
- Sequência: implementa os passos necessários para descrever qualquer programa;
 - **Exemplo**: sequências de passos em um fluxograma.



- Seleção: especifica a possibilidade de **selecionar** o fluxo de execução a partir de ocorrências lógicas;

A Linguagem de Programação C

- Iteração: permite a execução repetitiva de segmentos do programa.



Primeiro programa em C

- Um programa em **C** é um arquivo texto, chamado **código-fonte**, que contém a tradução de um algoritmo para a linguagem **C**;
- Este arquivo contém as declarações e as instruções do programa, escritas conforme a sintaxe da linguagem.

```
/*Meu primeiro programa em C*/
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Salvo no arquivo *hello.c*

Primeiro programa em C

Informa o compilador para incluir a biblioteca padrão de entrada/saída (stdio.h)

Comentário

```
/*Meu primeiro programa em C*/
```

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

```
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Função principal do programa

Salvo no arquivo *hello.c*

Função que imprime texto na tela

Primeiro programa em C

```
/*Meu primeiro programa em C*/
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Caractere especial que
Indica uma
“quebra de linha”

- Todas as instruções devem estar dentro das *chaves* que iniciam e encerram a função (“{” e “}”);
 - São executadas na ordem em que são escritas.
- As instruções em C sempre são encerradas com ponto-e-vírgula (“;”);
- Os **parâmetros** ou **argumentos** de uma instrução são passados dentro dos parêntesis (vide *printf()*).

Primeiro programa em C

- **Comentários** em um programa correspondem a porções de texto que são desconsideradas pelo compilador ou interpretador;
 - Servem para documentar o código-fonte;
- A linguagem C prevê **dois tipos** de comentários;
 - Comentários de múltiplas linhas (entre /* e */):

**/* Este comentário
pode ocupar mais de uma linha
do arquivo-fonte. */**
 - Comentários de uma única linha (iniciados com //):

// Este comentário deve estar contido em uma única linha

Primeiro programa em C

- Para executar o programa *hello.c*, o primeiro passo é *compilar* o código-fonte, para que o executável seja gerado;
 - O compilador é executado como qualquer outro programa;
- Supondo um ambiente **Linux/Unix** e o compilador *gcc* (*GNU C Compiler*), o programa anterior seria compilado e executado com os seguintes comandos:

Nome do arquivo-fonte
(entrada)

Nome do arquivo executável
(saída), indicado pela opção *-o*

```
$ gcc hello.c -o hello
```

```
$ ./hello
```

Hello, world!

Execução do programa

Chamada do compilador

Programas : Erros de Compilação

- Quando o programa não está de acordo com as regras da linguagem, o compilador retornará uma lista de *erros* e não gerará o arquivo executável;
- É importante **compreender as mensagens** de erro e **fazer as correções** no código-fonte.

```
/*Meu primeiro programa em C*/
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

```
$ gcc hello.c -o hello
hello.c: In function 'main':
hello.c:7: error: expected declaration or statement at end of input
```

Programas : Erros de Execução

- Quando o programa **não apresenta o comportamento esperado**, mesmo que a compilação não tenha resultado em erros, diz-se que ocorreu um ***erro de execução***;
- Alguns tipos de erros de execução:
 - A saída retornada não é a saída correta;
 - A execução do programa é encerrada abruptamente;
 - A execução não termina nunca (*loop infinito*);
 - ...
- Para auxiliar na localização de fontes de erros de execução, existem os ***depuradores (debuggers)***, que executam o programa passo a passo.

Exercícios

1. Escreva o programa *ex1.c* apresentado abaixo em um editor de textos (puro), compile-o usando o *gcc* e execute-o.

```
/*Meu primeiro programa em C*/
#include <stdio.h>

int main() {
    printf("Bem-vindo a disciplina SI100!\n");
    return 0;
}
```

Salvo no arquivo *ex1.c*

Exercícios

2. Modifique o programa anterior de forma a escrever cada palavra da frase “*Bem-vindo à disciplina SI100!*” usando uma chamada para a função ***printf()***.
3. Modifique o programa anterior para que **cada palavra** da frase “*Bem-vindo à disciplina SI100!*” seja escrita em uma linha.
4. Modifique o programa anterior para que cada letra da frase “*Bem-vindo à disciplina SI100!*” seja escrita em uma linha. Utilize **apenas** um comando ***printf()***.

ASPECTOS LINGUAGEM C

FUNDAMENTAIS

DA

(26)

Constantes

- Uma constante, como o próprio nome diz, tem valor *fixo* e *inalterável*.
 - Ex.:
 - ‘c’ – é um caractere simples;
 - 8 – número inteiro de valor 8;
 - ‘8’ – corresponde ao caractere 8 (e não ao número inteiro);
 - “*Primeiro programa*” – cadeia de caracteres (*string*);

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

O programa A imprime o número 2
O programa A imprime o número 2

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main() {
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

```
O programa A imprime o numero 2
```

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

```
O programa A imprime o numero 2
```

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

O programa A imprime o numero 2
O programa A imprime o numero 2
O programa A imprime o numero 2
O programa A imprime o numero 2
O programa A imprime o numero 2

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

```
O programa A imprime o numero 2
```

Constantes

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    printf("O programa A imprime o numero 2\n");
    printf("O programa %c imprime o numero 2\n",'A');
    printf("O programa A imprime o numero %d\n",2);
    printf("O programa %c imprime o numero %d\n",'A',2);
    printf("O programa %c %s %d\n",'A',"imprime o numero",2);

    return 0;
}
```

```
O programa A imprime o numero 2
```

VARIÁVEIS

Variáveis

- **Variáveis** são um aspecto fundamental em qualquer linguagem de programação;
- Correspondem a um *espaço de memória* reservado para **armazenar um certo tipo de dado**;
 - Contêm um **nome** para referenciar o seu conteúdo.
- Declarando uma variável em C:
 - `int num; // declara uma variável chamada "num", que
// armazena valores do tipo inteiro (números
// inteiros)`
- Atribuindo um valor a uma variável (já declarada):
 - `num = 3;`

Variáveis

- Exemplo de programa:

```
#include <stdio.h>

int main(){
    int num;
    num = 10;
    printf("O programa B imprime o numero %d\n", num);

    return 0;
}
```

- A saída deste programa será:

O Programa B imprime o numero 10

Tipos de Variáveis

- A linguagem C possui os seguintes tipos *básicos* de variáveis:
 - char: guarda um caractere;
 - int: guarda um número inteiro;
 - float: guarda um valor real com certa precisão;
 - double: guarda um valor real com precisão maior que a de float;
 - void: tipo vazio (sem valor).

Tipos de Variáveis

- Além dos 5 tipos básicos listados no slide anterior, existem também algumas variações:
 - **unsigned char**: caractere sem sinal (lembrar que um caractere pode ser interpretado como um número);
 - **long int**: número inteiro com domínio ampliado;
 - **unsigned int**: número inteiro sem sinal (positivo);
 - **unsigned long int**: número inteiro sem sinal com domínio ampliado;
 - **short int**: número inteiro com domínio reduzido;
 - **unsigned short int**: número inteiro sem sinal com domínio reduzido.

Tipos de Variáveis

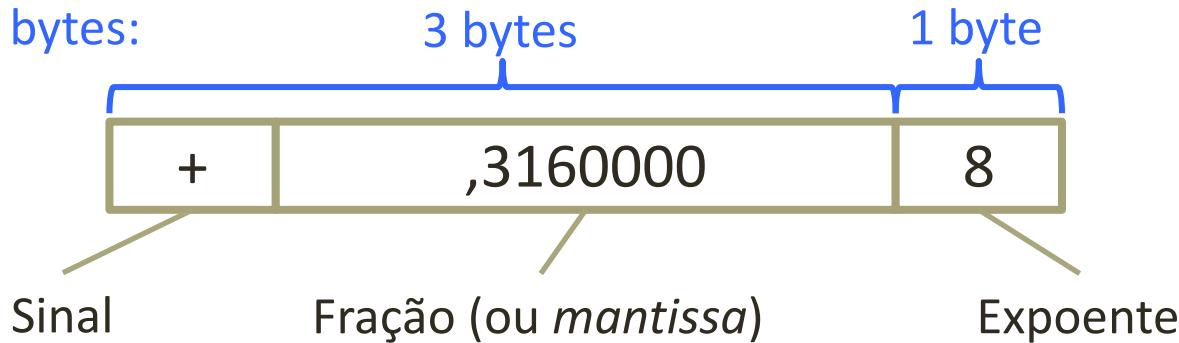
- As especificações de cada tipo de variável dependem da arquitetura do computador;
- Exemplo para um processador Pentium IV:

Tipo	Num de bits	Formato i/o	Início	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
int	32	%d	-2.147.483.648	2.147.483.647
unsigned int	32	%u	0	4.294.967.295
long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
float	32	%f	(+/-)10 ⁻³⁸	(+/-)10 ³⁸
double	64	%lf	(+/-)10 ⁻³⁰⁸	(+/-)10 ³⁰⁸

Variáveis de ponto flutuante

- Números em **ponto flutuante** (float, double) são usados para representar “números reais”;
- Em computador, números em ponto flutuante são representados de forma similar à **notação científica**:
 - Ex.: $+31.600.000 = +0,316 \times 10^8$
- Números em ponto flutuante são divididos em três partes:

Supondo 4 bytes:



Variáveis de ponto flutuante

- Como o número de bits disponível para representar a *mantissa* é finito, a precisão que se consegue representar em computador também é limitada.
 - São feitos “arredondamentos” em operações aritméticas;

```
#include <stdio.h>

int main() {
    float a = 1000.43;
    float b = 1000.0;
    printf("%f\n", a - b);
    return 0;
}
```

0.429993

Inicialização de variáveis

- A inicialização de uma variável pode ser feita tanto através de uma **atribuição** (posterior) quanto em sua **própria declaração**:

```
#include <stdio.h>

int main(){
    int evento ;
    char corrida;
    float tempo = 27.25;
    evento = 5;
    corrida = 'C' ;
    printf("O tempo vitorioso na eliminatoria %c",corrida);
    printf("\nA competicao %d foi %f.\n", evento, tempo);
    return 0;
}
```

O tempo vitorioso na eliminatoria C
da competicao 5 foi 27.250000.

Nomes de variáveis

- O **nome** de uma variável pode ser qualquer palavra que:
 - Não seja uma palavra reservada da linguagem:

```
auto      double    int       struct
break     enum      register  typedef
char      extern    return    union
const     float     short    unsigned
continue  for      signed   void
default   goto     sizeof   volatile
do        if       static   while
```

- Seja formado apenas por **letras, números e underline ('_')**;
- Não **inicie com número**;
 - **Ex.**: cAsA_dA_aNa33, _nome

Exercícios

5. Corrija o seguinte programa:

```
#include <stdio.h>
int main(){}
    printf(Existem %d semanas no ano., 56)
    return0;
)
```

6. Compile o programa abaixo, veja as mensagens de erro geradas e corrija-o.

```
#include <stdio.h>
int Main(void){
    int a=1; b=2; c=3;
    printf("Os numeros são: %d, %d e %d\n, a, b, c, d)
}
```

Exercícios

7. Dentre os nomes de variáveis abaixo, indique quais são aceitos pelo compilador da linguagem C e quais não são:

- 3ab
- ab3
- a3b
- FIM
- sim
- int
- \meu
- A
- nao
- A123
- papel-branco
- a*
- c++
- *nova_variavel

A importância dos nomes de variáveis

- O que faz o programa abaixo?

```
#include <stdio.h>

int main() {
    int abacaxi = 783;
    float acerola = 2500.00;
    float pitanga = 3750.00;
    float melancia = acerola + pitanga;

    printf("O valor calculado eh: %f\n", melancia);
    return 0;
}
```

A importância dos nomes de variáveis

- O que faz o programa abaixo?

```
#include <stdio.h>

int main() {
    int codigoDoFuncionario = 783;
    float salarioBase = 2500.00;
    float comissoesRecebidas = 3750.00;
    float salarioDoMes = salarioBase +
comissoesRecebidas;

    printf("O valor calculado eh: %f\n", salarioDoMes);
    return0;
}
```

O nome de cada variável deve indicar claramente sua função!

REFERÊNCIAS

Referências

- MIZRAHI, V. V., *Treinamento em Linguagem C – Curso Completo*, 2a Edição, Pearson Makron Books, 2005.
- MIYAZAWA, F. K., DE SOUZA, C. C. & KOWALTOWSKI, T.. *Notas de Aula da disciplina Algoritmos e Programação de Computadores*. Instituto de Computação, Unicamp.