Departamento
de Ciência da
COMPUTAÇÃO

UDESC
Joinville

Linguagem C
(introdução)

André Tavares da Silva

andre.silva@udesc.br

Como funciona um computador

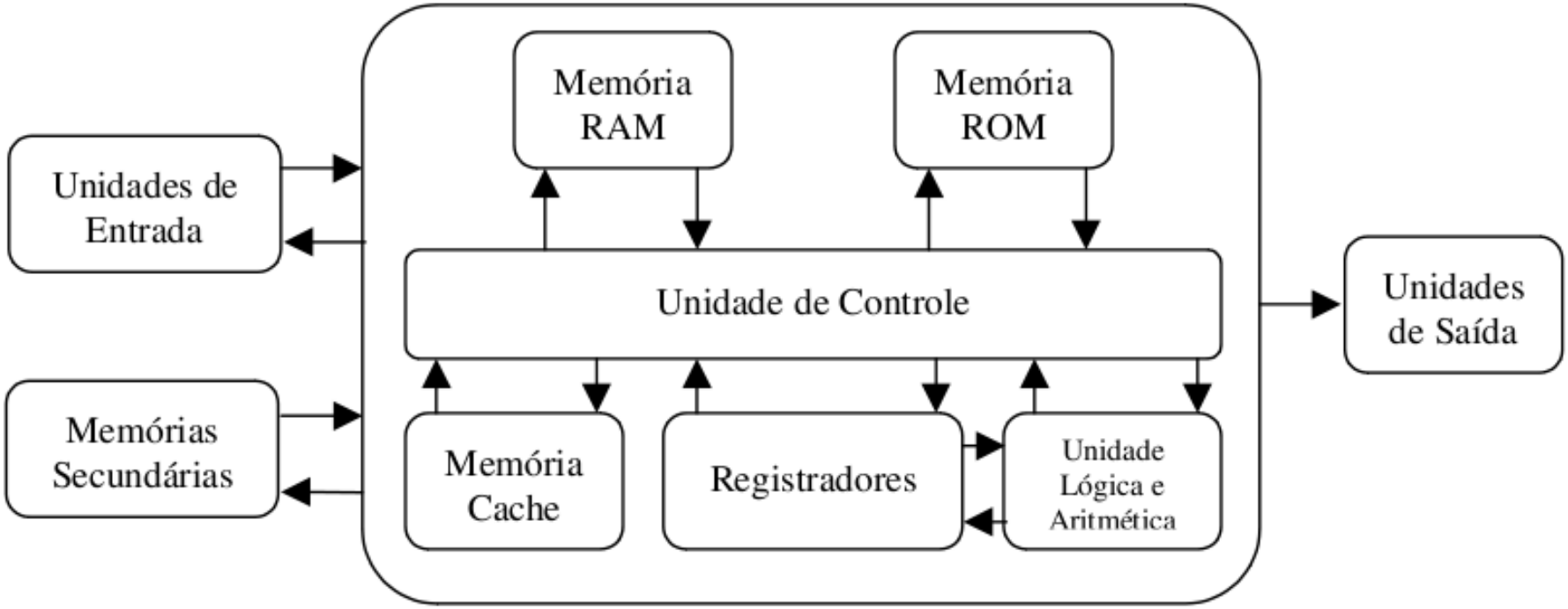


Figura: Organização Básica de um Computador Seqüencial (Miyazawa, 2001:1)

Ferramentas para programação de computadores

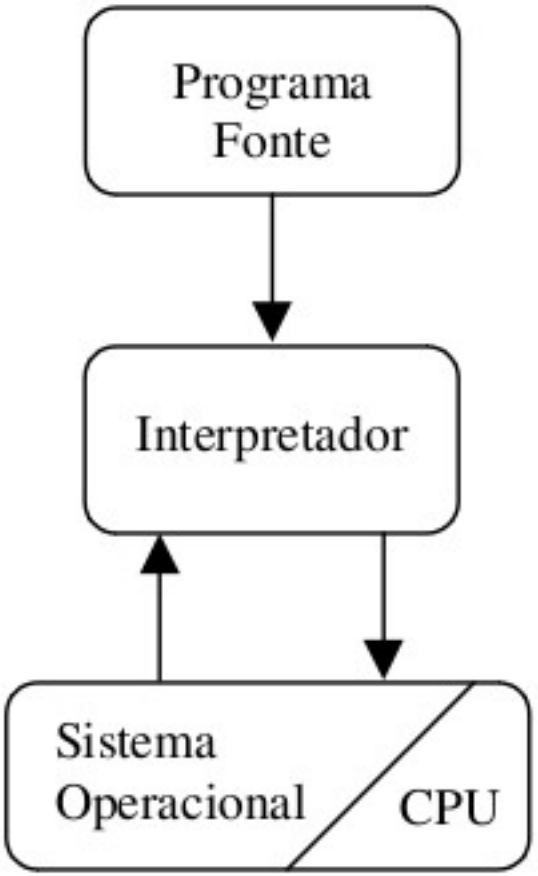
- **Pseudo-linguagem**: notação que se assemelha a uma linguagem de programação, mas que também possibilita ao programador concentrar-se no problema a ser modelado sem "se prender" a uma linguagem de programação específica. Essa notação mistura definições formais sobre dados e estruturas de controle, com informações em estilo livre.
- **Linguagem de programação**: uma linguagem desenvolvida para viabilizar a programação de computadores.
- **Ambiente de programação**: conjunto de tecnologias que dá suporte à programação de computadores (ex. Sistema Operacional, editor de texto, compilador, etc).

Linguagens

- **Linguagem de Máquina**: conjunto de instruções que podem ser interpretados e executados diretamente pela CPU de um dado computador. É específica para cada computador.
- **Linguagem Assembly** (Linguagem de Baixo Nível): Representação da linguagem de máquina através de códigos mnemônicos. Também é específica de cada máquina.
- **Linguagem de alto nível**: linguagem que independe do conjunto de instruções da linguagem de máquina do computador. Cada instrução de alto nível eqüivale a várias instruções da linguagem de máquina, sendo assim mais produtiva. Ex.: C, Delphy, Python, Lisp, Prolog, etc.

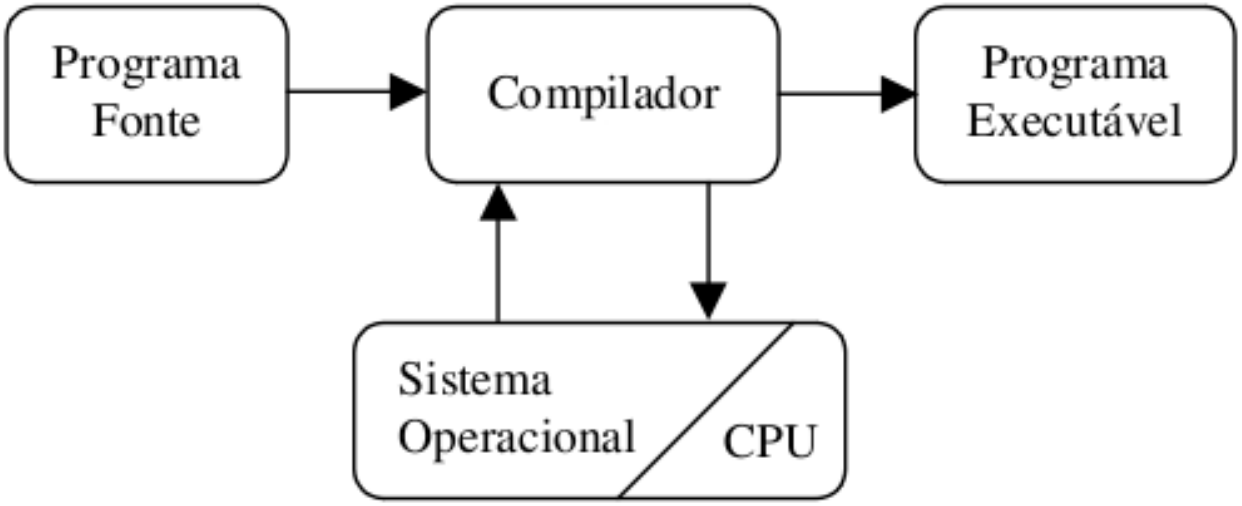
Interpretador

- É um programa que executa outros programas escritos em alguma linguagem de programação. Por outro lado, o uso de programas interpretados permite que trechos de código possam ser trocados por novos facilmente, fazendo com que o programa fonte possa mudar durante sua execução. Este é um dos grandes motivos de se usar programas interpretados em sistemas especialistas. Algumas linguagens para as quais podemos encontrar interpretadores são Javascript, Lua, Prolog, Python, entre outras.

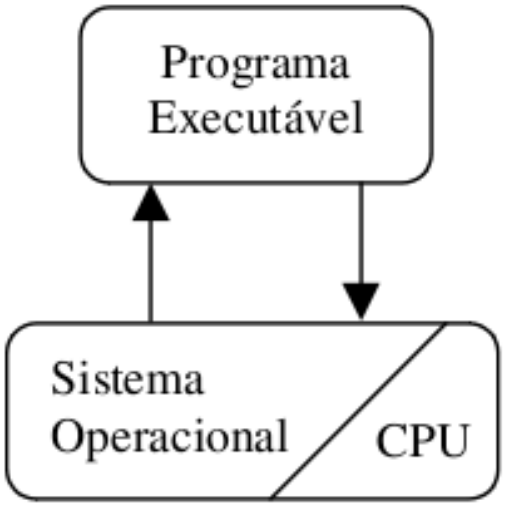


Compilador

- Tradutor de programas escritos em uma linguagem de programação para programas em linguagem de máquina (ex. GCC). Uma vez que o programa foi convertido para código de máquina, este pode ser executado independente do compilador e do programa original.



GERAÇÃO DO PROGRAMA EXECUTÁVEL



EXECUÇÃO DO PROGRAMA

Programa de computador

- A linguagem compreendida pelos computadores é a linguagem de máquina, cujo alfabeto é formado apenas por dois dígitos binários (ou bits): 0 e 1.
- Escrever programas em linguagem de máquina utilizando apenas 0s e 1s, entretanto, é uma tarefa árdua e bastante sujeita a erros.

Linguagens de baixo nível

- A linguagem de montagem (ou **assembly**) foi criada para facilitar a programação de computadores, mas ela ainda obriga o programador a escrever uma linha para cada instrução a ser executada pela máquina, forçando-o a raciocinar como máquina.
- O programa que traduz a linguagem de montagem para a linguagem de máquina é denominado montador (ou **assembler**).

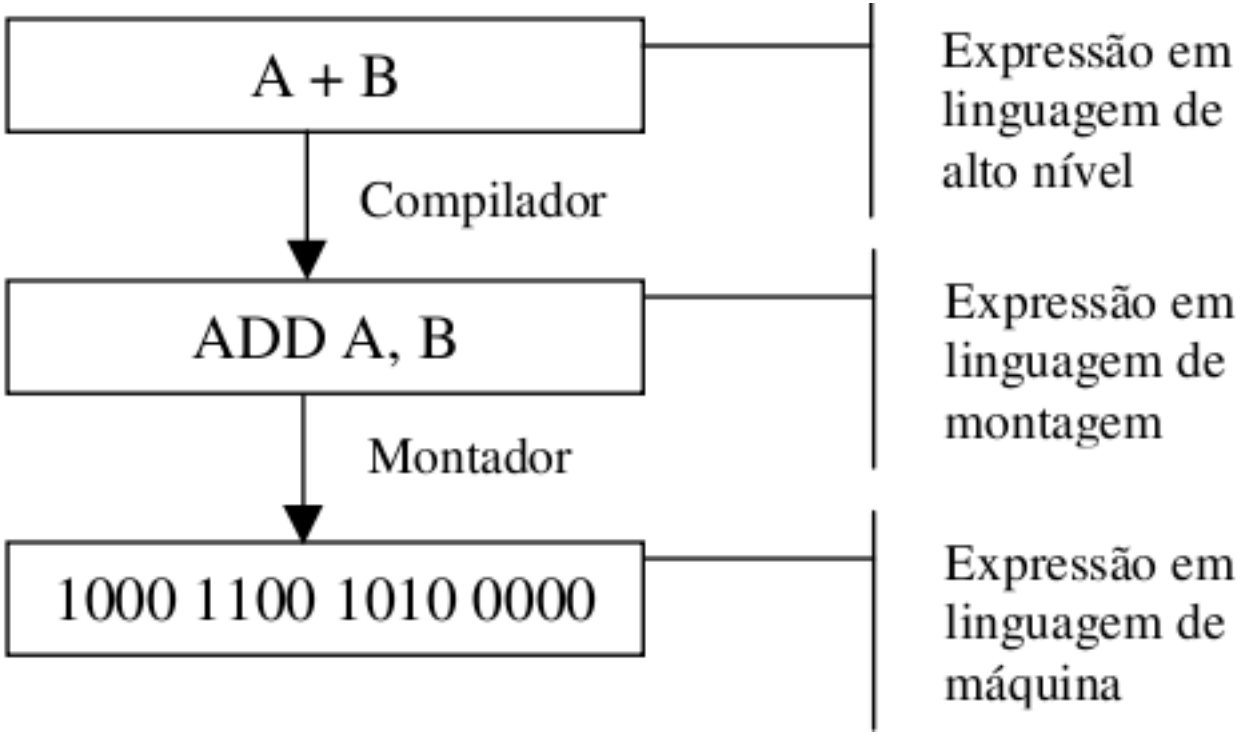
Linguagens de alto nível

- Já as linguagens de programação de alto nível oferecem uma série de vantagens ao programador:
  - Permitem que ele raciocine de uma forma mais natural, usando palavras em inglês e notações algébricas;
  - Colaboram para sua produtividade, por serem mais concisas que as linguagens de montagem;
  - Favorecem a portabilidade, uma vez que programas escritos em linguagens de alto nível são independentes do computador no qual foram desenvolvidos.

Linguagem C

- Atualmente, os programadores têm a sua disposição diferentes linguagens de programação de alto nível. A Linguagem C é uma delas que, em geral, é compilada. Portanto, como apresentado anteriormente, um programa codificado em linguagem C é traduzido pelo compilador em um programa executável codificado em linguagem de máquina. Às vezes essa tradução da linguagem de alto nível para a linguagem de máquina também passa por um montador, conforme apresentado na Figura 1, a seguir.

Linguagem C


A + B
Compilador
ADD A, B
Montador
1000 1100 1010 0000
Expressão em linguagem de alto nível
Expressão em linguagem de montagem
Expressão em linguagem de máquina

Linguagem C

- É uma linguagem de programação de propósito geral.
- Seus tipos de dados fundamentais são caracteres, inteiros e ponto flutuante de diversos tamanhos.
- Também oferece uma hierarquia de tipos de dados derivados criados com ponteiros, vetores, estruturas e uniões.
- Provê construções fundamentais de fluxo de controle exigidas por programas bem estruturados: agrupamento de comandos, tomada de decisão, laços, entre outros.
- Apresenta facilidades para modularização.

Tipos de Dados Primitivos

- São 5 os tipos básicos:
  - **char** - caractere
  - **int** - inteiro
  - **float** - ponto flutuante
  - **double** - ponto flutuante de precisão dupla
  - **void** - sem valor
- São 4 os modificadores:
  - **signed**
  - **unsigned**
  - **long**
  - **short**

Tipos de Dados Primitivos

Tipo	Tamanho	Menor valor	Maior valor
void	0	-	-
char	1	-128	127
unsigned char	1	0	255
int	4	-2.147.483.648	2.147.483.647
unsigned int	4	0	4.294.967.295
float	4	-3,4E-38	3,4E+38
double	8	-1,7E-308	1,7E+308

Identificadores

- Nomes utilizados para fazer referência às variáveis, funções, rótulos e vários tipos de objetos definidos pelo usuário.
- Podem ter de um a vários caracteres.
- Devem iniciar com letra ou sublinhado.
- Demais caracteres devem ser letras, números ou sublinhado.
- C diferencia letras maiúsculas de minúsculas.

Declaração de variáveis

- Posição nomeada de memória que é usada para guardar um valor que pode ser modificado pelo programa.
- Todas as variáveis devem ser declaradas antes de serem utilizadas.

<tipo> <identificador>

```
int i;

char a;
```

Constantes

- Uma constante é apenas um valor expresso literalmente no texto de um programa.
- Um número escrito explicitamente no programa é uma constante pois não irá modificar-se durante sua execução. Assim como os números escritos explicitamente, qualquer um dos tipos de dados podem ser constantes.
- O modificador const expressa que o valor de um identificador não pode ser alteradas durante o programa.
- Precede o tipo e o identificador: **const <tipo> <identificador>**

```
const char letra_A = 'A';
const double PI = 3.141592653;
```

Operadores Aritméticos

=	Atribuição
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)
--	Decremento
++	Incremento

operador=

```
x += valor;
y /= valor;
```

```
int v, w, i = 0;
v = ++i;
w = i++;
```

Operadores Relacionais

>	Maior que
>=	Maior que ou igual
<	Menor que
<=	Menor que ou igual
==	Igual
!=	Diferente

Operadores Lógicos

&&	E lógico (AND)
||	OU lógico (OR)
!	Negação (NOT)

Operadores Bit a Bit

&	E bit a bit (AND)
|	OU bit a bit (OR)
^	OR exclusivo (XOR)
~	Complemento de um
<<	Deslocamento à esquerda
>>	Deslocamento à direita

```
int v = 2;
v = v << 1;
v <<= 2;
```

Pré-Compilador C

- Trabalha sobre o programa antes de enviá-lo ao compilador.
- Substitui abreviações simbólicas existentes no programa pelas diretivas que representam.
- Todas as diretivas do pré-compilador começam com o símbolo #.

#include

- A diretiva #include indica uma solicitação de inserção de um outro arquivo no programa.

```
#include <stdio.h>
#include “meu_arquivo.h”
```

- stdio.h será procurado no diretório criado para arquivos de inclusão.
- meu_arquivo.h será procurado no diretório corrente.

Entrada e Saída

- Entrada/saída efetuadas por funções de biblioteca.
- Utilizar #include para incluir biblioteca.
- Biblioteca padrão *stdio.h*.

```
int printf (char *formato, ...);
int scanf (char *formato, ...);
int getchar (void);
int putchar (int c);
```

printf - formato

%[flags][comprimento][.prec]caractere_identificador_de_tipo

- [flags] - seqüência opcional de caracteres de sinalização (-, +, #, espaço em branco).
- [comprimento] - definição opcional de tamanho.
- [.prec] - especificador opcional de precisão (.0, .n).

```
int i = 0;
double pi = 3.1415;
printf("Inteiro: %d, double = %3.2f\n", i, pi);
```

formatos para printf/scanf

%c	escreve/lê um único caractere
%d	escreve/lê inteiro
%u	escreve/lê inteiro sem sinal
%f	escreve/lê ponto flutuante
%lf	escreve/lê double (*long float)*
%s	escreve/lê *string*
%o	escreve/lê octal
%x	escreve/lê hexadecimal

printf

```
int printf (char *formato, ...);
```

- Coloca na tela uma seqüência de caracteres.

```
int i = 0;
double pi = 3.14;

printf("Inteiro: %d, double = %f\n", i, pi);
```

scanf

```
int scanf (char *formato, ...);
```

- Lê vários tipos de dados.
- Lê somente até encontrar o primeiro separador (espaço em branco, tabulação ou nova linha).

```
int horas;
double valorHora;

printf("Entre: <horas trabalhadas> <valor hora>");
scanf("%d %f", &horas, &valorHora);
printf("A receber: %.2f\n", horas * valorHora);
```

scanf

- Lê todos os caracteres até o próximo caractere de espaço.
- Lê todos os caracteres até o primeiro que não puder ser convertido para o tipo especificado.
- Lê até n caractere, onde n é o tamanho especificado.
- Passar endereço (&) para *scanf* - passagem por referência - altera conteúdo do argumento.
- Strings lidas sem & - nome da string já é o endereço do primeiro elemento da string.

getchar

```
int getchar (void);
```

- Lê caracteres do teclado.
- Espera por um <ENTER>.
- Guarda <ENTER> no buffer.
- Pode “pular”caracteres.
- Mostra caractere na tela.

putchar

```
int putchar (int c);
```

- Escreve o argumento c na tela.

```
char c = 'g';


putchar( c );
putchar('S');
```

Exemplo

```
#include <stdio.h>

int main() {
    const float VALORHORA=10.0;

    int horas;
    float a_receber;


    printf("Entre numero de horas trabalhadas: ");
    scanf("%d", &horas );
    a_receber = horas * VALORHORA;
    printf("A receber: %.2f\n", a_receber);
    return 0;
}
```

Exemplo

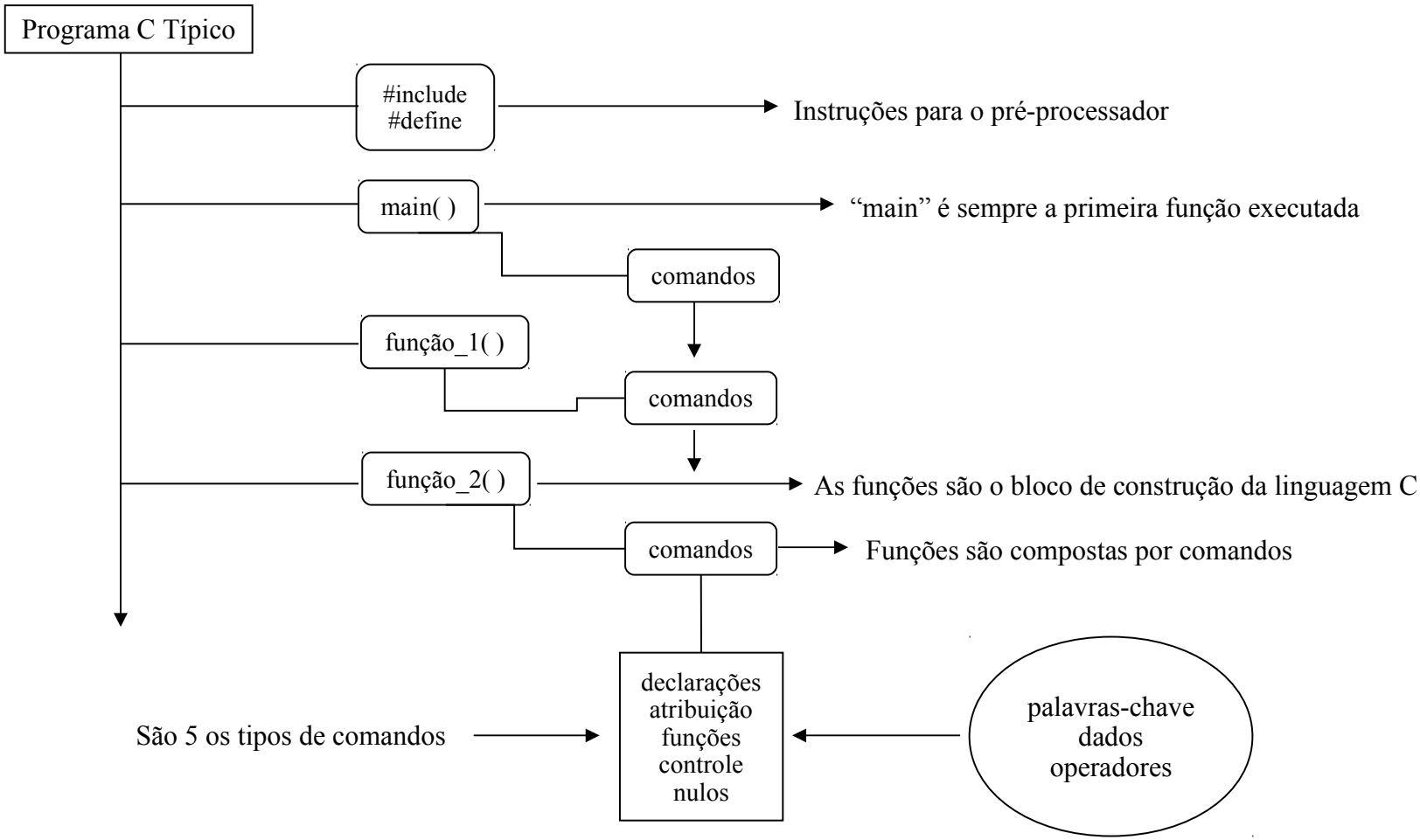
```
#include <stdio.h>          /* inclusão de biblioteca - entrada e saída */

int main() {                /* função principal */
    const float VALORHORA=10.0; /* definição de constante */

    int horas;                          /* variável inteira*/
    float a_receber;                    /* variável de ponto flutuante*/


    printf("Entre numero de horas trabalhadas: ");
    scanf("%d", &horas );
    a_receber = horas * VALORHORA;
    printf("A receber: %.2f\n", a_receber);
    return 0;
}
```

Estrutura de um Programa em C



Exercícios

- Compilar o programa anterior e executá-lo.
- Ler dois números quaisquer, calcular a soma, mostrar os números lidos e o resultado da soma.
- Ler dois números inteiros. Multiplicar o primeiro por 4 e o segundo por 0,6. Calcular a média aritmética dos resultados obtidos. Imprimir os valores lidos, os calculados e a média aritmética.

Exercícios

- Faça um programa para ler dois valores para as variáveis A e B, efetuando a troca dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresente ao final os valores trocados.
- Ler uma temperatura em graus Celsius e apresentá-la em graus Fahrenheit. A fórmula da conversão $F=(9*C+160)/5$, sendo F a temperatura em Fahrenheit e C a temperatura em graus Celsius.