

Linguagens de Programação

Linguagens de Programação

Uma linguagem de programação é um vocabulário e um conjunto de regras gramaticais usadas para escrever programas de computador. Esses programas instruem o computador a realizar determinadas tarefas específicas. Cada linguagem possui um conjunto único de palavras-chaves (palavras que ela reconhece) e uma sintaxe (regras) específica para organizar as instruções dos programas.

Os programas de computador podem ser escritos em várias linguagens de programação, algumas diretamente compreensíveis pelo computador e outras que exigem passos de tradução intermediária. As linguagens de programação podem ser divididas em três tipos, com relação à sua similaridade com a linguagem humana:

- Linguagem de máquina;
- Linguagem simbólica;
- Linguagem de alto nível.

Linguagens de Programação

Linguagem de máquina (machine language): é a linguagem de mais baixo nível de entendimento pelo ser humano e a única, na verdade, entendida pelo processador (UCP).

É constituída inteiramente de números (0's e 1's), o que torna praticamente impossível entendê-la diretamente. Cada UCP tem seu conjunto único de instruções que definem sua linguagem de máquina, estabelecido pelo fabricante do chip.

Uma instrução típica em linguagem de máquina seria algo como:

0100 1111 1010

Essa linguagem é também classificada como uma linguagem de primeira geração.

Linguagens de Programação

Linguagem simbólica (assembly): é a linguagem de nível imediatamente acima da linguagem de máquina. Ela possui a mesma estrutura e conjunto de instruções que a linguagem de máquina, porém permite que o programador utilize nomes (chamados mnemônicos) e símbolos em lugar de números.

A linguagem simbólica é também única para cada tipo de UCP, de forma que um programa escrito em linguagem simbólica para uma UCP poderá não ser executado em outra UCP de uma família diferente.

Nos primórdios da programação os programas eram escritos nessa linguagem.

Linguagens de Programação

Hoje a linguagem simbólica, é utilizada quando a velocidade de execução ou o tamanho do programa executável gerado são essenciais. A conversão da linguagem simbólica para a linguagem de máquina se chama montagem, e é feita por um programa chamado montador (ou assembler).

Uma típica instrução em linguagem simbólica seria:

ADD A, B

Essa linguagem é também classificada como linguagem de segunda geração, e, assim como a linguagem de máquina, é considerada uma linguagem de baixo nível.

Linguagens de Programação

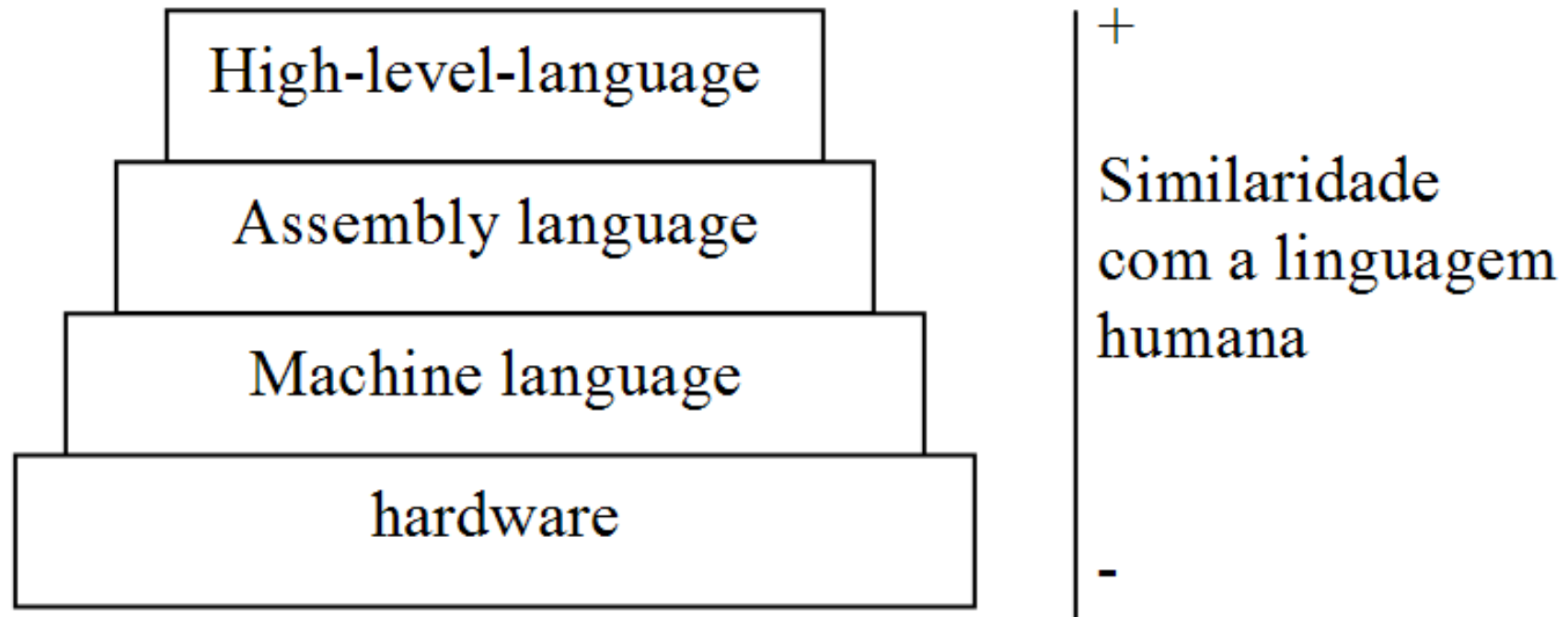
Linguagem de alto nível: São as linguagens de programação que possuem uma estrutura e palavras-chave que são mais próximas da linguagem humana. Tornando os programas mais fáceis de serem lidos e escritos. Esta é a sua principal vantagem sobre as linguagens de nível mais baixo.

Os programas escritos nessas linguagens são convertidos para a linguagem de baixo nível através de um programa denominado compilador ou de um interpretador.

Uma instrução típica de uma linguagem de alto nível é:

```
if (A>10) then A:=A-7;
```

Linguagens de Programação



Linguagem C

Breve histórico de “C”

- Criada por Dennis Ritchie;
- Em 1972;
- Centro de Pesquisas da Bell Laboratories;
- Para utilização no S.O. UNIX.

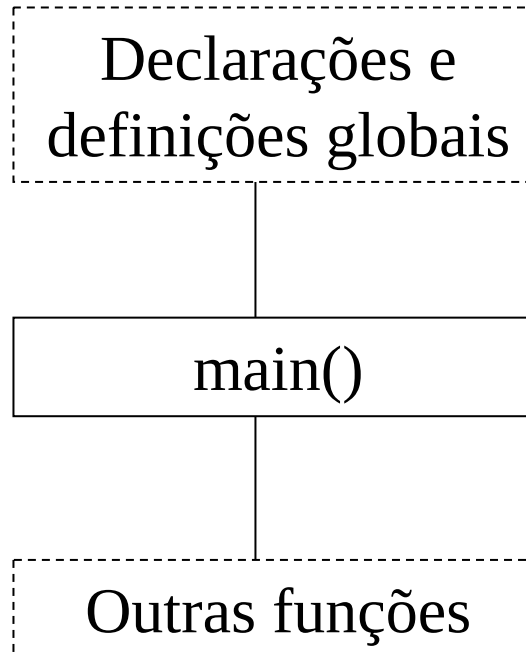
Características básicas da linguagem

- O C é uma linguagem de propósito geral;
- Sensível ao caso (Case sensitive);
- Tipos de dados primitivos: caractere, inteiro e real;
- Possui estruturas de controle de fluxo;
- Possui operadores aritméticos, lógicos, relacionais e condicional;
- Possibilita a modularização viabilizando a programação funcional;
- Todo programa tem uma função principal chamada **main()**;
- Toda linha de instrução em um programa é finalizada com um “;”.

ANSI

Devido à falta de padronização da linguagem C, em 1983, o **Instituto Norte-Americano de Padrões (ANSI)** formou um comitê, X3j11, para estabelecer uma especificação do padrão da linguagem C. O padrão foi completo em 1989 e ratificado como ANSI X3.159-1989 “Programming Language C” (**C ANSI**).

Estrutura de um programa em C



—— Obrigatório
----- Opcional

Conceitos Básicos – Linguagem C

Constantes

Exemplos:

- Decimal (10, -23768)
- Hexadecimal (0x12,0x1fea28)
- Octal (0123)
- Real (2.34, 2.34E+05, 2.14E-9)
- Caractere ('a', '%')

Palavras-reservadas

break	case	char	continue	do
double	else	float	for	if
int	return	sizeof	switch	struct
typedef	void	while	long	short
unsigned	signed			



Tipos Primitivos

➤ Caractere

- Definido pela palavra reservada char;
- Ocupa 8 bits (1 byte)
- Faixa de valores: -128 à 127
- Exemplo:

```
char letra;  
letra = 'A';
```


Tipos Primitivos

➤ Inteiro

- Definido pela palavra reservada int;
- Ocupa 16 bits (2 bytes)
- Faixa de valores: -32768 à 32767
- Exemplo:

```
int num;  
num = -73;
```

Tipos Primitivos

➤ Ponto flutuante e ponto flutuante de precisão dupla

- Definido pela palavra reservada float
- Ocupa 4 bytes
- Definido pela palavra reservada double
- Ocupa 8 bytes
- Faixa mínima de um valor em ponto flutuante
1E-37 a 1E+37
- Exemplo: float a,b,c=2.34;
double x=2.38,y=3.1415,z;

Operadores

A linguagem C disponibiliza, praticamente, o mesmo conjunto de operadores aritméticos, lógicos e relacionais apresentados anteriormente durante o estudo de algoritmos.

Operadores

➤ Operadores Aritméticos

➤ Unários: +, -, ++, --

Exemplos: +1

-5

a=-b;

Operador de atribuição

a++; $\Leftrightarrow$ a=a+1;

a--; $\Leftrightarrow$ a=a-1;

b=a++; $\Leftrightarrow$ b=a;

a=a+1;

b=++a; $\Leftrightarrow$ a=a+1;

b=a;

Obs.: a e b são variáveis numéricas.

Operadores

➤ Operadores Aritméticos

➤ Binários: +, -, *, /, %

Não existem os operadores ^ e \

Precedência (Hierarquia nas operações)

Hierarquia	Operação
1	Parênteses
2	Função
3	++,--
4	- (menos unário)
5	*, /, %
6	+, -

Operadores

► Operadores de Atribuição

=, +=, -=, *=, /=, %=

Exemplos: `a=5;`
 `a+=5; ⇔ a=a+5;`
 `a-=5; ⇔ a=a-5;`

Operadores

➤ Operadores Relacionais

Operador	Ação
>	maior que
>=	maior ou igual a
<	menor que
<=	menor ou igual
==	igual a
!=	diferente de

Operadores

► Operadores Lógicos

Operador	Ação
&&	e
	ou
!	não

Operadores

Precedência (Hierarquia dos operadores relacionais e lógicos)

Hierarquia	Operação
1	!
2	>, >=, <, <=
3	==, !=
4	&&
5	

Funções de Entrada e Saída Formatada

Conceito de função:

Ação, ato ou efeito.

Exemplo:

Dentre as funções dos vigias da UNIVASF, está a *função* de abrir a porta das salas de aula.

Contudo, para que um dos vigias abra a porta de uma das salas de aula temos que solicitar ao mesmo e informar o horário, dia e o número da sala a ser aberta.

Computacionalmente uma função representa uma seqüência de instruções que será executada para efetuar uma certa tarefa. Denominamos os dados necessários para que se execute uma determinada função como sendo os *parâmetros* da mesma.

Funções de Entrada e Saída Formatada

`#include <stdio.h>`

✚std → standard

✚io → input/output

printf ()

✚Forma geral:

printf (string_de_controle<,lista_de_argumentos>);

Funções de Entrada e Saída Formatada

➤ printf (continuação)

➤ *string_de_controle*

- descrição de tudo que a função colocará na tela;
- indica os caracteres;
- indica as variáveis com suas respectivas posições. Isso é feito, usando-se os códigos de controle, com a notação do %.

Funções de Entrada e Saída Formatada

Tabela de códigos de formato (%)

Código	Formato
%c	Um caractere (char)
%d	Um número inteiro decimal (int)
%i	O mesmo que %d
%e	Número em notação científica com o "e"minúsculo
%E	Número em notação científica com o "e"maiúsculo
%f	Ponto flutuante decimal
%g	Escolhe automaticamente o melhor entre %f e %e
%G	Escolhe automaticamente o melhor entre %f e %E
%o	Número octal
%s	String
%u	Decimal "unsigned" (sem sinal)
%x	Hexadecimal com letras minúsculas
%X	Hexadecimal com letras maiúsculas
%%	Imprime um %

Funções de Entrada e Saída Formatada

► printf (continuação)

✦ *lista_de_argumentos*

Para cada código % contido na string de controle, temos um argumento correspondente na *lista_de_argumentos*

Funções de Entrada e Saída Formatada

► printf (continuação)

Vamos ver alguns exemplos:

Código	Imprime
<code>printf ("Um %%%c indica %s",'c',"char");</code>	Um %c indica char
<code>printf ("%X %f %e",107,49.67,49.67);</code>	6B 49.670000 4.967000e+001
<code>printf ("%d %o",10,10);</code>	10 12

Exercício

Construa um programa que escreva a string “juros de”, o inteiro 10 e o caractere ‘%’ na tela, constituindo a seguinte frase:

juros de 10%