Slides trabalhados durante a quinta aula prática

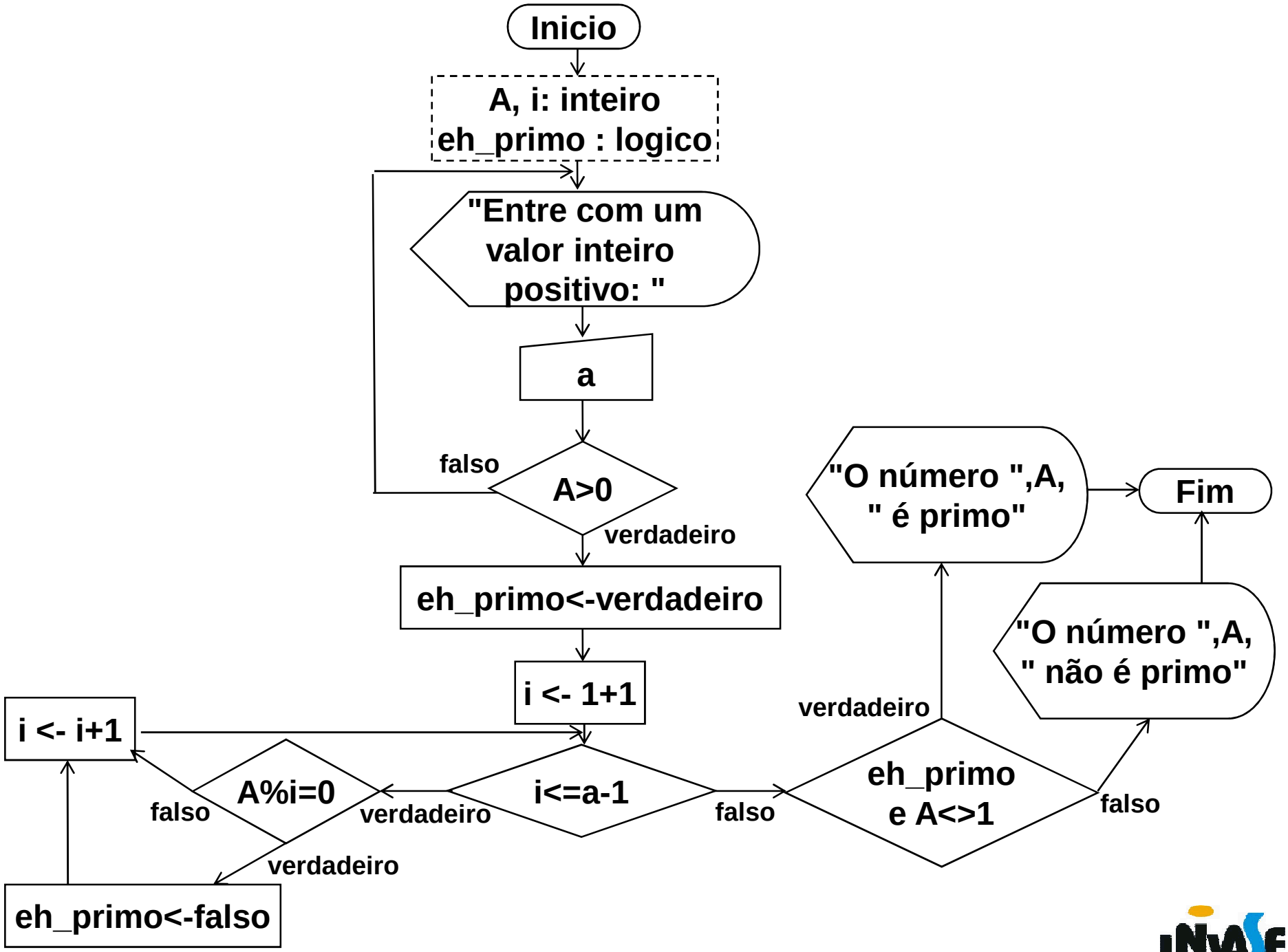
3. Laços de repetição (continuação)

Exercício:

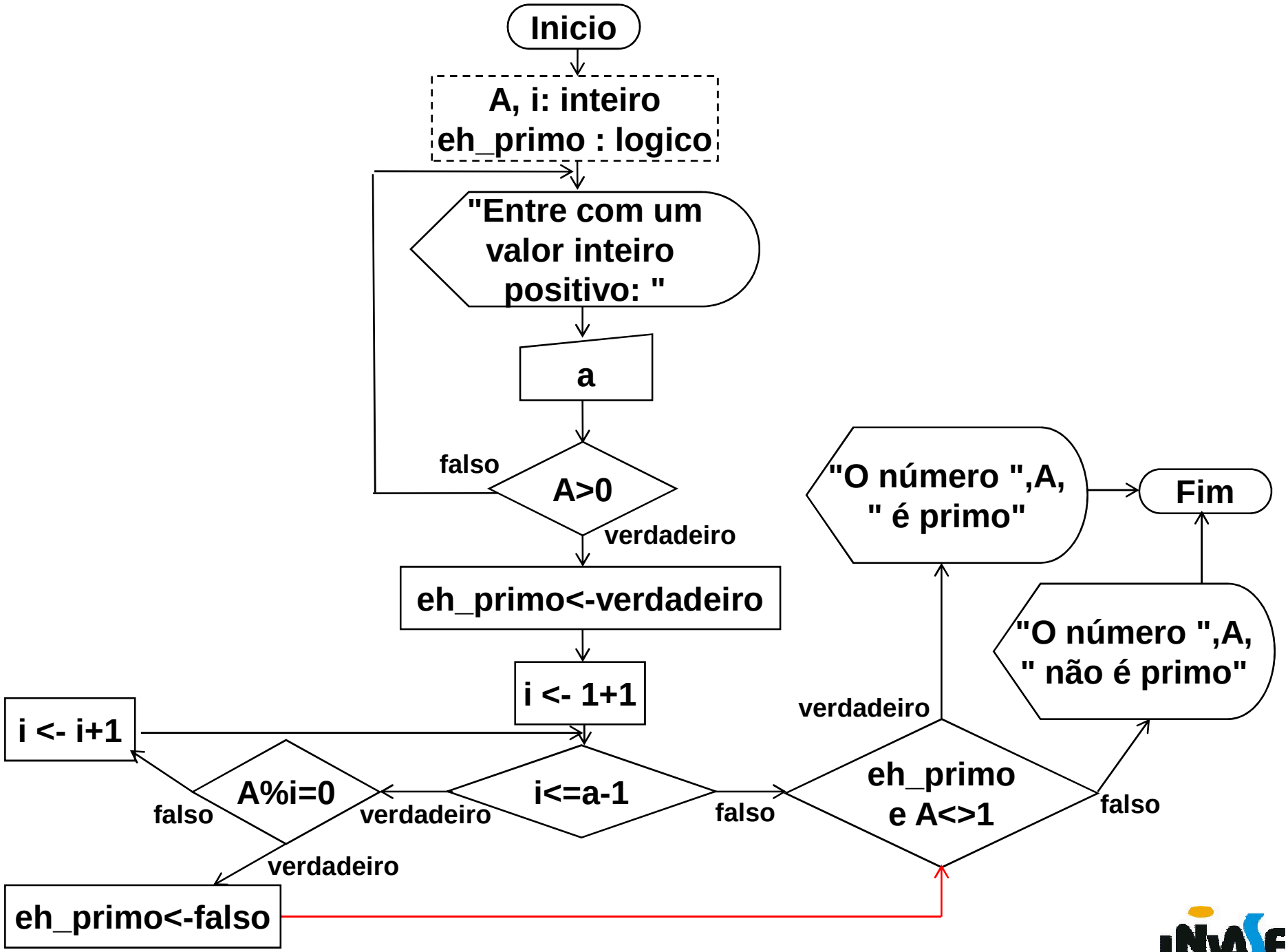
Construa um algoritmo, representando-o através de um pseudocódigo e de um fluxograma, que leia da entrada padrão um número inteiro positivo e retorne na saída padrão uma mensagem indicando se o número lido é ou não um número primo. As entradas devem ser validadas.

```
algoritmo "exercício"
var    A, i, numeroDeDivisoes : inteiro
inicio
   repita
      escreva ("Entre com um valor inteiro positivo: ")
      leia (A)
   ate (A>0)
   numeroDeDivisoes <- 0
   para i de 1 ate A faca
      se (A%i=0) entao
       numeroDeDivisoes <- numeroDeDivisoes + 1
      fimse
   fimpara
   se (numeroDeDivisoes = 2) entao
        escreva ("O número ",A," é primo")
   senao
        escreva ("O número ",A," não é primo")
   fimse
fimalgoritmo
```

```
algoritmo "exercício"
var    A, i: inteiro
         eh_primo: logico
inicio
   repita
      escreva ("Entre com um valor inteiro positivo: ")
      leia (A)
   ate (A>0)
   eh_primo <- verdadeiro
   para i de 1+1 ate A-1 faca
      se (A%i=0) entao
           eh_primo <- falso
      fimse
   fimpara
   se (eh_primo e A<>1) entao
        escreva ("O número ",A," é primo")
   senao
        escreva ("O número ",A," não é primo")
   fimse
fimalgoritmo
```


Inicio
A, i: inteiro
eh_primo : logico
"Entre com um valor inteiro positivo: "
a
A>0
falso
verdadeiro
eh_primo<-verdadeiro
i <- 1+1
i <- i+1
A%i=0
falso
verdadeiro
verdadeiro
i<=a-1
falso
eh_primo<-falso
eh_primo e A<>1
verdadeiro
falso
"O número ",A, " é primo"
"O número ",A, " não é primo"
Fim

```
algoritmo "exercício - usando interrompa"
var    A, i: inteiro
       eh_primo: logico
inicio
  repita
    escreva ("Entre com um valor inteiro positivo: ")
    leia (A)
  ate (A>0)
  eh_primo <- verdadeiro
  para i de 1+1 ate A-1 faca
    se (A%i=0) entao
        eh_primo <- falso
        interrompa //causa uma saída imediata do laço
    fimse
  fimpara
  se (eh_primo e A<>1) entao
     escreva ("O número ",A," é primo")
  senao
     escreva ("O número ",A," não é primo")
  fimse
fimalgoritmo
```


Inicio
A, i: inteiro
eh_primo : logico
"Entre com um valor inteiro positivo: "
a
A>0
falso
verdadeiro
eh_primo<-verdadeiro
i <- 1+1
i <- i+1
A%i=0
falso
verdadeiro
verdadeiro
i<=a-1
falso
eh_primo e A<>1
verdadeiro
falso
"O número ",A, " é primo"
"O número ",A, " não é primo"
Fim
eh_primo<-falso

Slides trabalhados durante a sexta aula teórica

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 languale): 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

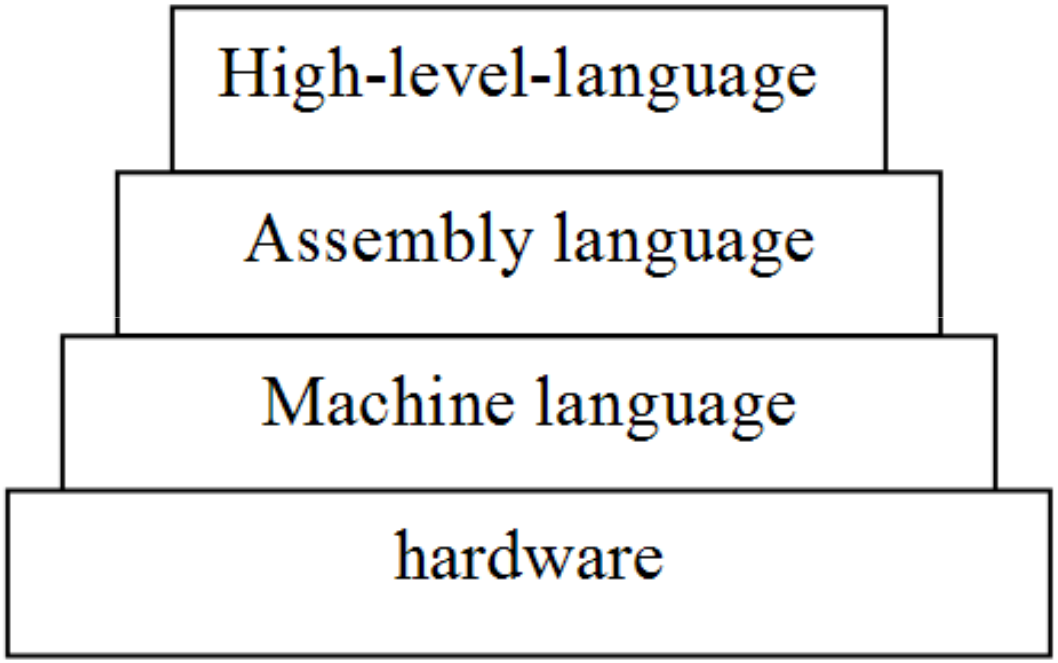
Linguagem de alto nível: São as linguagens de programação que possuem uma estrutura e palavras-chave que são mais próximas da linguagem humana. Tornando os programas mais fáceis de serem lidos e escritos. Esta é a sua principal vantagem sobre as linguagens de nível mais baixo.

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



+

Similaridade com a linguagem humana

-

Linguagem C

Breve histórico de “C”

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

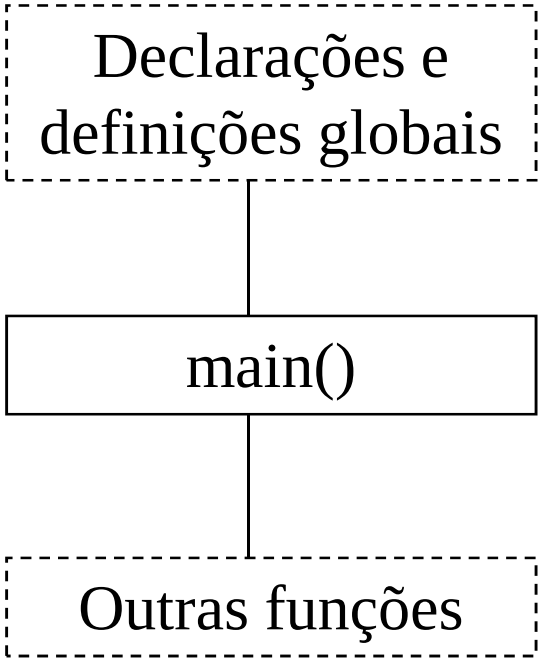
Características básicas da linguagem

- 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 e relacionais;
- Possibilita a modularização viabilizando a programação imperativa ou procedural;
- Todo programa tem uma função principal chamada **main()**;
- Todo 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

- Palavras Reservadas

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

- Comentários
  - /* ccccccccccccccccccccccccccccccccccccccccc
    ccccccccccccccccccccccccccccccccccccccccc*/

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++; ⇔ a=a+1;

a--; ⇔ a=a-1;

b=a++; ⇔ b=a;

a=a+1;

b=++a; ⇔ a=a+1;

b=a;

Obs.: a e b são variáveis numéricas.