Linguagem C

1 - Declaração de variáveis

```
/* Portugol */

INTEIRO: a, b, c;
REAL: d, e;
CARACTER: f;
LÓGICO: g;
```

```
/* Linguagem C - A linguagem C não possui nativamente a representação para valores lógicos (bool). Para usar uma váriável lógica, faz-se necessário a inclusão da biblioteca <stdbool.h> */

int a, b, c;
float d, e;
char f;
bool g:
```

2 - Operadores

2.1 - Operador de atribuição

O operador de atribuição é o sinal "=". Seu uso em programação é ligeiramente diferente de seu uso na matemática normal. Se escrevermos

x = y;

em um programa em C, isto significa que o valor de y deve ser atribuído em x, e não que y é igual a x como seria de se esperar. Em uma instrução de atribuição, o lado direito pode ser qualquer expressão e o lado esquerdo deve necessariamente ser o nome de uma variável. A sintaxe, portanto, é:

variavel = expressao;

O operador de atribuição também pode ser usado de forma encadeada:

bananas = tomates = laranjas = 50;

Nesse exemplo, será atribuído o valor 50 às variáveis bananas, tomates e laranjas.

2.2 - Operadores aritméticos

Operador	Símbolo	Ação	Exemplo
Adição	+	Soma seus dois operandos	x + y
Subtração	−	Subtrai o segundo operando do primeiro operando	x − y
Multiplicação	*	Multiplica seus dois operandos	x * y
Divisão	/	Divide o primeiro operando pelo segundo operando	x / y
Módulo	%	Fornece o resto da divisão do primeiro operando pelo segundo operando	x % y

2.3 - Operadores relacionais

Operador	Símbolo	Pergunta respondida	Exemplo
Igual	==	Operando 1 é igual ao operando 2?	x == y
Maior que	>	Operando 1 é maior que o operando 2?	x > y
Menor que	<	Operando 1 é menor que o operando 2?	x < y
Maior ou igual a	>=	Operando 1 é maior ou igual ao operando 2?	x >= y
Menor ou igual a	<=	Operando 1 é menor ou igual ao operando 2?	x <= y
Diferente	!=	Operando 1 é diferente do operando 2?	x != y

2.4 - Operadores lógicos

Operador	Símbolo	Verdadeiro quando	Exemplo
E	&&	Expressão 1 **E** expressão 2 são verdadeiras	exp1 && exp2
OU	||	Expressão 1 **OU** expressão 2 são verdadeiras	exp1 || exp2
Não	!	A expressão é falsa	!exp1

3 - Funções de entrada e saída

```
/* Portugol */

ESCREVA("Informe número: ");
LEIA(num);

ESCREVA("O dobro de ", num, " é igual a ", num * 2);
```

```c
/* Linguagem C - É utilizada as seguintes especificações de formato para entrada e saída: %d (números inteiros), %f (números decimais), %c (char) e \n para quebra de linha. */

printf("Informe número: ");
scanf("%d", &num);

printf("O dobro de %d é igual a %d", num, num * 2);
```

4 - Estruturas condicionais

4.1 - Seleção simples

```
/* Portugol */

SE(condição) ENTÃO
  bloco_de_comandos_1;
SENÃO
  bloco_de_comandos_2;
FIM_SE
```

```c
/* Linguagem C */

if(condição){
  bloco_de_comandos_1;
}
else{
  bloco_de_comandos_2;
}
```

4.2 - Seleção composta

```
/* Portugol */

ESCOLHA (variável)
  CASO valor_1: bloco_de_comandos_1;
  CASO valor_2: bloco_de_comandos_2;
  CASO valor_3: bloco_de_comandos_3;
  ...
  CASO valor_n: bloco_de_comandos_n;
  CASO CONTRARIO: bloco_de_comandos_específico;
FIM_ESCOLHA
```

```c
/* Linguagem C */

switch (variável){
```

```
  case valor_1: bloco_de_comandos_1;
  break;
  case valor_2: bloco_de_comandos_2;
  break;
  case valor_3: bloco_de_comandos_3;
  break;
  ...
  case valor_n: bloco_de_comandos_n;
  break;

  default: bloco_de_comandos_específico;
}
```

5 - Estruturas de repetição

5.1 - Repetição com teste no início

```
/* Portugol */

ENQUANTO(condição) FAÇA
  bloco_de_comandos;
FIM_ENQUANTO
```

```
/* Linguagem C */

while(condição){
  bloco_de_comandos;
}
```

5.2 - Repetição com teste no fim

```
/* Portugol */

REPITA
  bloco_de_comandos;
ENQUANTO(condição);
```

```
/* Linguagem C */

do{
  bloco_de_comandos;
}while (condição);
```

5.3 - Repetição com variável de controle

```
/* Portugol */

PARA(inicialização; condição; atualização) FAÇA
  bloco_de_comandos;
FIM_PARA
```

```
/* Linguagem C */

for(inicialização; condição; atualização){
  bloco_de_comandos;
}
```

6 - Vetores

```
/* Portugol */

/* Declaração de um vetor com capacidade para 5 números inteiros */
INTEIRO: numeros[5];

/* Inserção do valor 15 na primeira posição do vetor */
numeros[0] ← 15;

/* Recuperação de elemento do vetor e atribuição a uma variável */
x ← numeros[3];
```

```
/* Linguagem C */

/* Declaração de um vetor com capacidade para 5 números inteiros */
int numeros[5];

/* Inserção do valor 15 na primeira posição do vetor */
numeros[0] = 15;

/* Inserção de valores no vetor durante a sua declaração */
int numeros[5] = {15, 20, 25, 30, 35};

/* Inserção de valores no vetor durante a sua declaração, sem definição de tamanho do
vetor */
int numeros[] = {15, 20, 25, 30, 35};

/* Recuperação de elemento do vetor e atribuição a uma variável */
x = numeros[3];
```

7 - Matrizes

```
/* Portugol */

/* Declaração de uma matriz com 2 linhas e 3 colunas */
INTEIRO: numeros[2][3];

/* Inserção do valor 15 na primeira posição da matriz */
numeros[0][0] ← 15;

/* Recuperação de elemento da primeira posição da matriz e atribuição a uma variável */
x ← numeros[0][0];
```

```
/* Linguagem C */

/* Declaração de uma matriz com 2 linhas e 3 colunas */
int numeros[2][3];

/* Inserção do valor 15 na primeira posição da matriz */
numeros[0][0] = 15;

/* Inserção de valores na matriz durante a sua declaração */
int numeros[2][3] = {{15, 20, 25}, {30, 35, 40}};

/* Recuperação de elemento da primeira posição da matriz e atribuição a uma variável */
x = numeros[0][0];
```