Linguagem C

Linguagem C

- **Uma linguagem difundida:**
  - Amplamente utilizada...
  - Uma linguagem veterana...
  - Sintaxe difundida, servindo como inspirações tecnológicas.
  - Linguagem clássica
- **Uma linguagem multi-nível:**
  - Permite compor programas com abordagens variando entre ‘baixo e alto nível’
- **Organização**:
  - Funções e estruturas de informação.
- **Ponteiros:**
  - Permite a independência de memória pré-alocada.

Linguagem C

- Devido a sua **flexibilidade de utilização**, ela pode ser considerada como complicada:
  - Uma alternativa é o uso de outras linguagens “menos flexíveis”, i.e. especializada para a aplicação em questão.
  - Entretanto, o fato é que compreender C, com suas flexibilidades, permite compreender outras linguagens.

Estrutura de um Programa em C

```
/*Diretivas de Pré-processamento*/
#include ....
#define ....
```

```
/*Declarações Globais*/
/*Protótipos de Funções*/
```

```
/* Função principal – marca o início da execução do programa*/
int main( ) {
        declarações locais;
        comandos;
         ....
}
```

```
/*Funções*/
Tipo função1 (declaração de parâmetros){
     /*declarações locais;*/
     /*comandos;*/
}
```

Tudo em um arquivo .c

Tradução – Portugol para C

```
Início /*Calculo do perímetro e a área

        de uma circunferência*/

    inteiro R;
    real Perm, Area, PI;

    PI ← 3.14159;
    Escreva(“Valor do raio:”);
    Leia(R);
    Perm ← 2 * PI * R;
    Area ← PI*R**2;
    Escreva(“Perímetro:”);
    Escreva(R);
    Escreva(“Área: “)
    Escreva(Area);
Fim;
```

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

int main(void){

        int R;
        float Perm, Area, PI;

        PI = 3.14159;
        printf(“Entre com o valor do raio:”);
        scanf(“ %d”,&R);
        Perm = 2 * PI * R;
        Area = PI* pow(R,2);
        printf(“Perímetro: %.2f\n”,Perm);
        printf(“Área: %.2f\n”,Area);
}
```

Tipos de Dados

- **char (caractere) :** armazena caracteres ou números literais
  - Ex: ‘a’ ‘1’ ‘\n’
- **int (inteiro)** : armazena números inteiros
  - Ex. 50, 017
- **float (real)** : armazena números com ponto flutuante em precisão simples (até 6 ou 7 dígitos significativos dependendo da máquina)
  - Ex: 6.5 -12.4 1.2e-3 -0.00013
- **double (real)** : armazena números com ponto flutuante em precisão dupla (até 15 ou 16 dígitos significativos dependendo da máquina)
  - Ex: 0.51231298967 -1.3e-15

Modificadores de Dados

- **unsigned** : armazena número sem sinal (positivo)
  - Ex unsigned int
- **short** : reduz os limites de variação
  - Ex . short int
- **long** : amplia os limites de variação
  - Ex. long int
- **void** : tipo usado em funções para indicar parâmetros ou valores de retorno vazio

Tabela de Dados

Tipo	Bytes	Leitura e Escrita	Intervalo	
			Inicio	Fim
char	1	%c	-128	127
unsigned char	1	%c	0	255
signed char	1	%c	-128	127
int	2	%i	-32.768	32.767
unsigned int	2	%u	0	65.535
signed int	2	%i	-32.768	32.767
short int	2	%hi	-32.768	32.767
unsigned short int	2	%hu	0	65.535
signed short int	2	%hi	-32.768	32.767
long int	4	%li	-2.147.483.648	2.147.483.647
signed long int	4	%li	-2.147.483.648	2.147.483.647
unsigned long int	4	%lu	0	4.294.967.295
float	4	%f	3,4E-38	3.4E+38
double	8	%lf	1,7E-308	1,7E+308
long double	10	%Lf	3,4E-4932	3,4E+4932

Expressões em C

- **As expressões em C envolvem normalmente:**
  - **Constantes, Variáveis e Operadores;**
  - **Ex:** Perm = 2 * PI * R;

    Area = PI* pow(R,2);

    Retorno = ((A > B) && (C<0));

Constantes em C

- Representam valores fixos inteiros ou caracteres
- **Constantes Inteiras**
  - Algarismos decimais (0 – 9)
  - Atenção: *não iniciam por 0* Ex. 10 -98 1000005
- **Constantes de Ponto Flutuante**
  - Constante em precisão simples (float) : 0.023F 1.2e-4F 3.4f
  - Constante em precisão dupla (double): -0.5e-15 1.24e-8 1.2
- **Constantes Alfanuméricas**
  - Caractere representado entre apóstrofes
  - Ex: ‘a’ ‘:’ ‘\n’ ‘2’
- **Constantes “string”**
  - Seqüência de caracteres entre aspas
  - Ex. “maria” “Av. Sete de Setembro”

Variáveis em C

- As variáveis armazenam informações que podem ser alteradas ao longo do programa.
- Todas as variáveis devem ser declaradas antes de serem usadas.

- **Declaração de Variáveis:**
  - A declaração de uma variável envolve a definição do *tipo* e do *identificador*
  - A declaração da variável associa uma área reservada na memória (total de bytes - depende do tipo declarado) ao nome da variável (identificador).

  - Ex: int QLAT; char Sexo; float area;

Variáveis em C

- **Regras para Identificação de Variáveis:**
  - O identificador deve começar por
  - Letra: ‘a’ a ‘z’ ou ‘A’ a ‘Z’ ou ‘_’ (sublinhado)
  - A seguir podem vir letras, ‘_’ e algarismos de 0 a 9
  - Ex: Total_de_latas, _x1, Nota2, nota2
- **Maiúsculas ≠ Minúsculas (*Case-Sensitive)***
  - Ex: int Idade, int idade, int idadE;

    float Peso, float peso;

- **Não são permitidos nomes ou palavras reservadas:**
  - auto break case char const continue default do double else enum extern float for goto if int long main register return short signed sizeof static struct switch typedef union unsigned void volatile while

Operadores em C

- Operador de Atribuição:
  - Ex: A =10;

```
/*Exemplo*/

float B;
int a,b,c,d;
B = 2.4F;
a=b=c=d=1;
```

- Operadores Aritméticos:
  - Soma: +
  - Subtração: -
  - Multiplicação: *
  - Divisão: /
  - Resto da divisão: %

```
/*Exemplo*/
...
float X;
int A,B,C,D;
A=10;
B=15;
C = A+B-2;
D = B%A;
X = B/A;
X = B/2;
X= B/2.0f;
```

Operadores em C

- Operadores Relacionais:
  - Maior: >
  - Menor: <
  - Maior ou igual: >=
  - Menor ou igual: <=
  - Igual: ==
  - Diferente: !=

```
/*Exemplo*/

5 > 10
4 < 2

‘a’ == 97

char a,b;
a = ‘c’;
b = ‘c’;
a == b
‘a’ == ‘b’
```

Operadores em C

- **Operadores Lógicos**:
  - Interseção (e) : &&
  - União (ou): ||
  - Negação (não): !
- São usados para combinar expressões relacionais compostas
- Resultam em 0 (falso) e >0 (verdadeiro)

```
/*Exemplo*/

int contador =1;
char letra = ‘s’;
!( (contador <= 10) && (letra == ‘p’) )
(letra ==’s’) || (letra == ‘S’)
```

Funções de Entrada/Saída

- **A função printf:**
- Sintaxe da chamada
  - ***printf("expressão de controle", lista de argumentos);***
- Expressão de controle
  - caracteres impressos na tela + cód. de formatação dos argumentos;
- Lista de argumentos:
  - expressões, variáveis ou constantes;
- Exemplo:
  - Algoritmo: Escreva("O valor da variável é"); Escreva(Y);
  - Em C: printf("O valor da variável é %d\n", Y); /*se Y é int*/

ou

printf("O valor da variável é %f\n", Y); /*se Y é float*/

Funções de Entrada/Saída

- **A função scanf**
- Sintaxe da chamada
  - ***scanf("expressão de controle", lista de argumentos);***
- Expressão de controle:
  - caracteres lidos do teclado + cód. de formatação dos argumentos
- Lista de argumentos:
  - endereços das variáveis a serem lidas
- Operador de Endereço &:
  - Exemplo : &Nfilhos
- Exemplo:
  - Algoritmo: leia(A,B);
  - Em C: scanf("%d %d”,&A, &B); /*se A e B são int */

ou

scanf("%f %f”,&A, &B); /*se A e B são float */