Linguagem de Programação I

Tipos de Dados – Aula 09

Profª Ms. Lília Marta Brandão Soussa Modesto

Tipos de Dados em C

1. Dados Numéricos:

- Tipos Inteiros: nºs positivos e negativos.

Tipo de dado inteiro	Faixa de abrangência
int	de –32.768 a 32.767
long int	de –2.147.483.648 a 2.147.483.647
unsigned int	de 0 até 65.535
unsigned long int	de 0 até 4.292.967.295

Tipos de Dados em C

1. Dados Numéricos:

- Tipos Reais: nºs positivos e negativos e fracionários.

Tipo de dado real	Faixa de abrangência
float	de 3.4 e–38 até 3.4 e+38
double	de 1.7 e–308 até 1.7 e+308
long double	de 3.4 e–4932 até 1.1 e+4932

Tipos de Dados em C

2. Dados Caracteres:

- São sequências contendo letras, números e símbolos especiais.

Tipo de dado caracter	Faixa de abrangência
char	de 0 até 255 caracteres

Tipos de Dados em C

3. Dado Vazio ou Vácuo:

- Utilizado para declarar funções que não retornam valor.

Tipo de dado	Faixa de abrangência
void	Vazio ou Vácuo – sem valor

Variável

- Deve ser identificada pelo seu tipo e por um nome.
- Em C, deve ser declarada antes de ser usadas.
- Forma geral: tipo lista_de_variáveis.

Exemplos:

```
int CONT = 1;
float  SB, HT;
char LETRA  = 'a';
```

Variável – Regras

1) Nome com até 32 caracteres;

2) O 1º caractere deverá ser uma letra ou sublinhado;

3) Não poderá possuir espaços em branco;

4) Não poderá ser palavra reservada a uma palavra_chave de C, não deve ter o mesmo nome que as funções criadas ou que estão na biblioteca de C;

5) Só poderão ser letras, números e o caractere sublinhado "_";

6) Há diferença entre caracteres maiúsculos e minúsculos. Então NOME, nome, Nome, noME são diferentes.

Variável – Regras

Padronizar:

- Começar com uma letra;
  - Utilizar sempre letras maiúsculas;
  - Variáveis com no máximo 10 caracteres.

Tipos de Variáveis

1) Variáveis Locais: São declaradas dentro de funções. Não são reconhecidas por outras funções.

Padronizar: Devem ser declaradas no início do bloco, antes de qualquer comando.

```
void func1(void)
 {
     int i;
     int j;
     i = 10;
     j = 30;
 }
```

```
void func2(void)
    {
        int x;
        x = 10;
     }

void func3()
    {
        int x;
        x = -199;
     }
```

Tipos de Variáveis

2) **Parâmetros Formais**: são declaradas na definição dos parâmetros das funções que usam argumentos. Elas receberão os valores dos argumentos. São utilizadas para que uma função reconheça as variáveis de outra função. Suas declarações ocorrem depois do nome da função e dentro do parênteses. Se comportam como qualquer variável local dentro da função.

```
int soma(int n1, int n2)
  {
    int s;
    s = n1 + n2;
    return s;
  }
```

A função soma() tem dois parâmetros: n1 e n2. Essa função retorna o valor da soma.

Tipos de Variáveis

3) **Variáveis Globais**: são declaradas fora de todas as funções. São reconhecidas pelo programa inteiro. Guardam seus valores durante a execução do programa. Porém, se uma variável global e uma local possuem o mesmo nome, todas as referências ao nome da variável dentro do bloco onde a variável local foi declarada dizem respeito à variável local e não tem efeito sobre a global.

Padronizar: Devem ser declaradas no início do programa, antes da função principal.

Tipos de Variáveis

```
#include <stdio.h>
int cont;

void func1(void);
void func2(void);

void main (void)
{
   cont = 100;
   func1();
}

void func1(void)
{
   int temp;
   temp = cont;
   func2();
}

void func2(void)
{
   int cont = 0;
   for (cont; cont<10; cont++)
        printf (´.´);
}
```

1º Inclui bibliotecas (funções pré-definidas da linguagem)

2º Declara variáveis globais

3º Declara funções do programa

4º Programa principal (main)

Variável

Se o conteúdo da variável do tipo caractere tiver mais de uma letra, então a variável será um vetor e cada letra ocupará uma posição.

```
char NOME [5];
```

	0	1	2	3	4
NOME =	F	E	L	I	Z

NOME [0] = F

NOME [1] = E

NOME [2] = L

NOME [3] = I

NOME [4] = Z

Variável

float NOTA [8], MD [8] [4], NUM;

	0	1	2	...	7
NOTA =	4.5	6.5	8.0		6.0

NOTA [0] = 4.5

NOTA [1] = 6.5

NOTA [2] = 8.0

...

NOTA [7] = 6.0

MD =

	0	1	2	3
0				
1				
2				
...				
7				

MD[0][0], MD[0][1], ... MD[7][3]

e NUM é uma variável simples

Constante String

- String é um conjunto de caracteres colocado entre aspas duplas (“ ”).

  Exemplo: printf (“\nOlá Pessoal”);

Constante Caractere de Barra Invertida

Código	Significado
\n	Desce para uma nova linha

Operadores Aritméticos

Operador	Operação
+ +	Incremento
– –	Decremento
*	Multiplicação
/	Divisão
%	Resto Divisão Inteiro
+	Adição
–	Subtração ou inversão de sinal

Operadores

Atribuição Múltipla: x = y = z = 0;

Incremento: x = x + 1;
é o mesmo que: ++x;
x +=1;

Decremento: x = x – 1;
é o mesmo que: x– –;
x –=1;

OBS: ++x é diferente de x++
– –x é diferente de x – –

Ex: x = 10;
y = x++;
} y = 10, x = 11

Ex: x = 10;
y = ++x;
} x = 11, y = 11

Operadores

Qual o valor final de A, B, X e Y?

A = 4; B = 8;

X = A--;
```
X=A;
A=A-1;
```

Y = --A;
```
A=A-1;
Y=A;
```

X+=B++;
```
X=X+B;
B=B+1;
```

Y -=++B;
```
B=B+1;
Y=Y-B;
```

Operadores

Qual o valor final de A, B, X e Y?

```
A = 4; B = 8;

X = A--;


Y = --A;


X+=B++;


Y -=++B;
```

Operadores

A	B	X	Y
4	8	4	2
3	9	**12**	**-8**
2	**10**		

Operadores

Qual o valor final de A, B, X e Y?

```
A = 10; B = 7;

A -= B++;

X = ++A;

X += --B;

Y  =++B;

A *= (X+9);
```

Operadores

A	B	X	Y
10	7	4	8
3	8	**11**	
4	7		
80	**8**		

Operadores Relacionais

Operador	Ação
==	Igual a
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

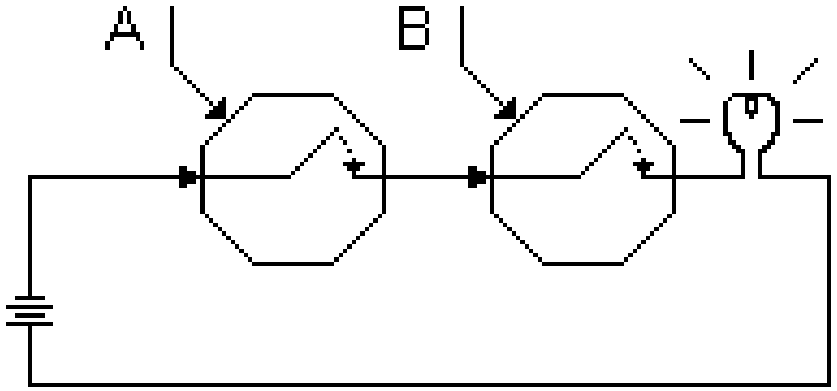
Operadores Lógicos

Operador	Ação
&&	AND
| |	OR
!	NOT

Operador Lógico && (AND)

O operador **AND** produz uma saída 1, se todas as entradas forem 1.

Entradas		Saída
A	B	&&
0	0	0
0	1	0
1	0	0
1	1	1



Operador Lógico && – Exercícios

1) Seja A=0110 e B=1101.
Calcule A && B

A	B	A && B
0	1	
1	1	
1	0	
0	1	

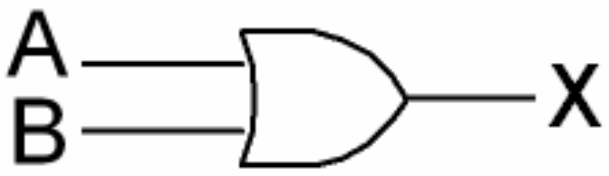
2) Seja A=0101,
B=0011,
C=1111.
Calcule A && B && C

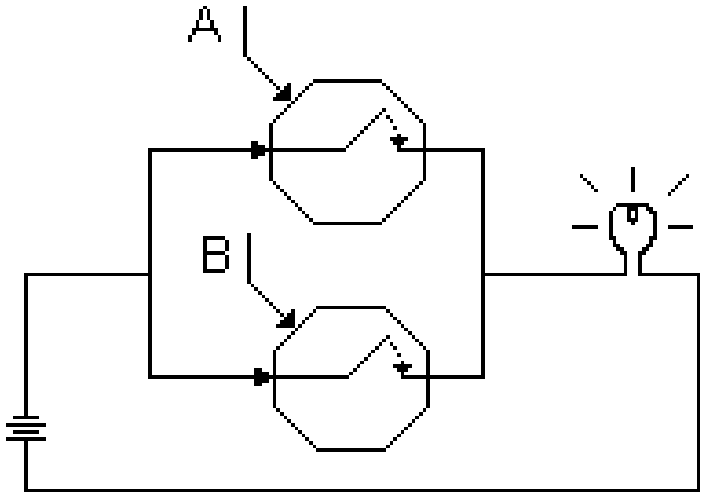
A	B	A&&B	C	A&&B&&C
0	0		1	
1	0		1	
0	1		1	
1	1		1	

Operador Lógico || (OR)

O operador **OR** produz uma saída 1, se pelo menos uma entrada for 1.

Entradas		Saída
A	B	||
0	0	0
0	1	1
1	0	1
1	1	1





Operador Lógico || – Exercícios

1) Seja A=0110 e B=1110.
   Calcule A || B

A	B	A||B
0	1	
1	1	
1	1	
0	0	

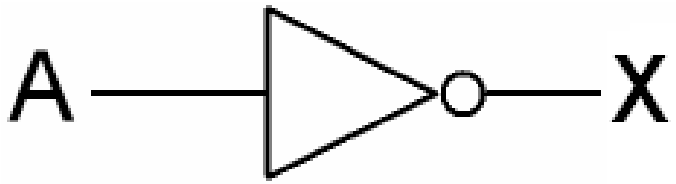
2) Seja A=1100,
   B=1111,
   C=0001.
   Calcule X=A || B || C

A	B	A||B	C	A||B||C
1	1		0	
1	1		0	
0	1		0	
0	1		1	

Operador Lógico ! (NOT)

É chamado de **inversor** ou **função complemento**. O operador **NOT** inverte o valor da entrada, produzindo na saída o valor oposto.

Entrada	Saída
A	!
0	1
1	0



Operadores Lógicos – Exercícios

1) Calcule X = A && B !! C

A	B	A&&B	C	X
0	1		1	
1	1		0	
1	1		0	
0	0		1	

2) Calcule X = A && B || !C

A	B	A&&B	C	!C	X
1	1		1		
1	0		0		
0	1		0		
0	1		1		

Operadores Lógicos

A expressão :

```
10 > 5 && !(10 < 9) || 3 <= 4
```

é verdadeira ou falsa?

Operadores Lógicos

10 > 5 && !(10 < 9) || 3 <= 4

1º) (10 < 9) ⇨ F

2º) ! (F) ⇨ V

3º) 10 > 5 ⇨ V

4º) 3 <= 4 ⇨ V

5º) V && V ⇨ V

6º) V || V ⇨ V

Expressões Aritméticas

Cálculo da Área de um triângulo	
Fórmula Matemática	$\text{ÁREA} = \frac{\text{BASE} \cdot \text{ALTURA}}{2}$
Expressão Aritmética	AREA = (BASE * ALTURA) / 2;

Cálculo da Área da Circunferência	
Fórmula Matemática	$\text{ÁREA} = \pi \cdot \text{RAIO}^2$
Expressão Aritmética	AREA = PI * pow(RAIO,2);

Fórmula Matemática	$X = \{3 \cdot [20 : (3 + 2)]\}$
Expressão Aritmética	X = (3 * (20 / (3 + 2)));

Expressões Aritméticas

Exercício:

Qual o resultado das expressões:

a) X = (4 * (6 / (2 +1)))

b) X = (4 * (6 / 2 +1))

c) X = (4 * 6 / (2 +1))

d) X = (4 * 6 / 2 +1)

e) X = 2 * 5 % 3 + 8

Expressões Aritméticas

a) X = (4 * (6 / (2 +1)))

6 / 3

4 * 2

8

b) X = (4 * (6 / 2 +1))

3 +1

4 * 4

16

c) X = (4 * 6 / (2 +1))

24 / 3

8

d) X = (4 * 6 / 2 +1)

24 / 2

12 +1

13

e) X = 2 * 5 % 3 + 8

10 % 3

1 + 8 = 9

Palavras reservadas

- Toda palavra reservada (palavra-chave) é escrita em letra minúscula.

Palavras-chave de C			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Atenção:

else é uma palavra reservada, mas **ELSE** não.

Programa C

- Todo programa em C consiste em uma ou mais funções.
- A única função que necessariamente precisa estar presente é **main()**, que é a primeira função a ser chamada quando a execução do programa começa.

Exemplo:

```
/* Programa de Boas Vindas */
#include <stdio.h>
#include <stdlib.h>
int main (void)
{
   printf ("\nBem Vindos\n");
   system("pause");
   return (0);
}
```

Comentário: vem entre **/*** e ***/**