PROGRAMAÇÃO I

Prof. Luiz Albano

luiz.albano@ifsp.edu.br

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Bragança Paulista

Agenda

Tema: Introdução Linguagem C

- Tipos de dados
- Variáveis
- Formato de Dados
- Comando de saída
- Comando de entrada
- Comentários

Comando de Saída

Comando para enviar uma frase para a saída (monitor).

Comando de Saída

Comando: printf()

A função **`printf( )`** é uma função de saída de dados.

Permite que uma mensagem seja exibida na tela do computador.

As mensagens devem ser escritas **ENTRE ASPAS (“”)**.

Para utilizarmos necessitamos da biblioteca de funções stdio.h. Fazemos a inclusão desta biblioteca no início do código fonte através da seguinte declaração:

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

Comando de Saída

Exemplo:

```
#include <stdio.h>
main() {
    printf(“Ola Mundo!”);
}
```

Saída:
Ola Mundo!

Comando de Saída

Caracteres Especiais

O símbolo **** é utilizado para remover o significado especial que o caractere posterior representa

Exemplo: no caso das aspas (“), retira o significado do delimitador de strings (conjunto de caracteres).

Caractere Especial	Função / Representação
\n	Nova linha
\t	Tabulação
\f	Salto de página
\a	Sinal sonoro
\r	Retorna o cursor no início da linha
\\	Barra invertida
\0	Caractere nulo
\’	Aspa simples
\”	Aspas duplas

Comando de Saída

Exemplo:

```
#include <stdio.h>
main() {
    printf(“Linha1 Linha2”);
    printf(“\nIFSP\nBraganca Paulista\n”);
    printf(“Tecnico Mecatronica”);
}
```

Saída:

Linha1 Linha2
IFSP
Braganca Paulista
Tecnico Mecatronica

Comando de Saída

Formatos de Dados

Sempre que quisermos escrever um tipo de dado dentro de um **printf()** devemos substituir este valor indicando um formato de escrita. A tabela abaixo apresenta os códigos de formatos de dados para os tipos que trabalhamos na linguagem C.

Formato	Função
%d	Usado quando, no texto, for necessário exibir um número inteiro
%f	Usado quando, no texto, for necessário exibir um número com casas decimais.
%c	Usado quando, no texto, for necessário exibir um caractere
%s	Usado quanto, no texto, for necessário exibir uma sequencia de caracteres.

Comando de Saída

Exemplo:

```
#include <stdio.h>
main() {
    printf(“Valor: R$ %f”, 29.99);
    printf(“\nIdade: %d”, 25);
    printf(“\nTurma: ”, ‘D’);
    printf(“\nCurso: ”, “Mecatronica”);
}
```

Saída:
Valor: R$ 29.99
Idade: 25
Turma: D
Curso: Mecatronica

Comentários

Quando desenvolvemos programas, devemos colocar textos que expliquem o raciocínio seguido durante seu desenvolvimento para que outras pessoas, ou nós mesmos, ao ler o programa mais tarde, não tenhamos dificuldades em entender sua lógica. Esses textos são chamados de comentários.

Os comentários podem aparecer em qualquer lugar do programa. Em C, há dois tipos de comentários: os **comentários de linha** e os **comentários de bloco**.

Comentários

Comentários de Linha

Os comentários de linha são identificados pelo uso de **//**. Assim, quando usamos **//** em uma linha, tudo o que estiver nessa linha depois do **//** são considerados comentários.

Exemplos:

```
//isto é um comentário

int x=0; //a partir daqui temos um comentário
```

Comentários

Comentários de Bloco

Os comentários de bloco são iniciados por **/*** e finalizados por ***/**. Tudo o que estiver entre esses dois símbolos são considerados comentários. Os comentários de bloco podem ocupar várias linhas.

Exemplo:

```
/*
Isto é um comentário de bloco
Podemos continuar escrevendo até que seja
declarado
*/
```

Comentários

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
  int matricula= 2233;  /* podemos escrever cometários desta forma */
  float media_final=80.5; // ou apenas com duas barras no início do comentário
  char discipl[10]="Prog I";// a var discipl[10], pode armazenar até 10 caracteres
  printf ("O aluno matricula = %d \n", matricula);
  printf ("Disciplina = %s \n", discipl);
  printf ("Ficou com media = %f \n\n", media_final);
  system("pause");
  return(0);
}
```

Tipos de Dados e Variáveis

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Bragança Paulista

Variáveis

Sempre que desejarmos armazenar um valor que, por qualquer razão, não seja constante (fixo), devemos fazê-lo utilizando variáveis.

Um variável nada mais é do que um nome que damos a uma determinada posição de memória para conter um valor de um determinado tipo.

Como o próprio nome indica, o valor contido em uma variável pode variar ao longo da execução de um programa.

Uma variável deve ser sempre definida antes de ser utilizada. A definição de uma variável indica ao compilador qual o tipo de dado que fica atribuído ao nome que indicarmos para esta variável.

Variáveis

Definindo uma variável

Para podermos utilizar variáveis, temos a obrigação de reservar esta variável na memória do computador.

```
TipoDados nomeVariavel;
TipoDados nomeVariavel1, nomeVariavel2;
```

Onde:

TipoDado: deve ser substituído pelo tipo de dado que será armazenado

nomeVariavel: deve ser substituído pelo nome que você deseja chamar sua variável

Variáveis

Tipos de Variáveis

Para declarar uma variável devemos especificar o tipo de informação que ela pode armazenar. A tabela abaixo apresenta os tipos utilizados pela linguagem C.

Tipo	Tamanho	Intervalo
char	8 bits	-128 a 127
int	32 bits	-2.147.483.648 a 2.147.483.647
float	32 bits	$3,4x10^{-38}$ a $3,4x10^{38}$
void	0 bits	sem valor

Variáveis

Declarando uma variável

Os nomes de variáveis (identificador), devem ter como primeiro caractere letra. Os demais podem ser letras, números ou o caractere underline (_).

Em C há **distinção entre caracteres maiúsculos e minúsculos**. Assim, para evitar erros desse tipo, aconselhamos evitar o uso de caracteres maiúsculos nos nomes das variáveis, apesar de seu uso ser permitido.

Exemplo de declaração:

```
int numero;
```

Variáveis

Declarando uma variável

Todas as variáveis devem ser declaradas no início do programa.

O fato de declarar a variável no início não significa que ela está inicializada.

Exemplo:

```
#include <stdio.h>
main() {
    int contador;
    float preco_do_quilo;
    char letra_digitada;
    int idade_Manoel, idade_joao, idade_maria;
}
```

Variáveis

Atribuindo valores (guardando um valor)

Sempre que declaramos uma variável, estamos solicitando ao compilador para reservar espaço em memória para armazená-la. Este espaço passa a ser referenciado através do nome da variável.

O tipo de dado determina a quantidade de bytes que que ela ocupará na memória.

Sempre que uma variável é definida, um conjunto de bytes fica associado a ela. Dessa forma, quando uma variável é criada fica automaticamente com um valor aleatório que resulta da disposição dos bits que se encontram nos bytes reservados para a representação dessa variável (o que chamamos de valor de lixo).

Variáveis

Atribuindo valores (guardando um valor)

Quando uma variável é declarada fica sem com um valor, o qual resulta do estado aleatório dos bits que a constituem.

Desse modo, uma variável poderá ser iniciada com um valor através de uma operação de atribuição.

A atribuição de um valor só pode ser realizada para variáveis. Ao realizar uma atribuição o valor anterior presente na variável é eliminado, ficando nela o novo valor que lhe foi atribuído.

Variáveis

Atribuindo valores (guardando um valor)

Uma atribuição de valor à uma variável é realizada obedecendo a seguinte sintaxe:

```
variável = expressão;
```

A atribuição pode ser realizada em qualquer ponto do programa, após a definição da variável.

Uma variável pode ser automaticamente iniciada quando se faz a sua declaração.

Variáveis

Exemplos:

```
char sexo = ‘F’;
/*a variável sexo do tipo char foi declarada e
recebeu o valor F (note as aspas simples)*/

float salario = 552.35;
/*a variável salario foi declarada e recebeu o
valor 552,35*/

salario = 1625.23;
/*a variável salario recebeu o valor 1.625,23
(note que o separador de decimais é o ponto “.” e
não a vírgula como utilizamos naturalmente no
Brasil!*/
```

Variáveis

Exemplos:

```
//Cálculo da área de um retângulo
#include <stdio.h>
main() {
    float base, altura, area;
    base = 8.3;
    altura = 2.5;
    area = base * altura;
    printf("Area retangulo: %f", area);
}
```

Entrada de Dados

Comando para receber valores digitados pelo usuário, através do teclado.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Bragança Paulista

Entrada de Dados (Leitura de Dados)

Atribuindo valores (guardando um valor)

Comando: `scanf("%formato", &var)`

A função **`scanf( )`** é responsável por ler os dados que forem digitados pelo teclado e o armazenar na variável indicada pelo segundo parâmetro.

O **"%formato"**, deve ser substituído pelo formato de leitura da informação. Para ler dados de outros tipos serão utilizados outros códigos, conforme veremos a seguir.

E o valor **&var** deve ser substituído pelo nome da variável que irá armazenar o valor digitado pelo usuário. Exemplo: &altura.

Entrada de Dados (Leitura de Dados)

Formatos de Dados

Os códigos de formatação servem para indicar na funçã scanf() o formato do dado que será lido, ou seja, que o usuário deverá digitar na tela.

Formato	Função
%d	Usado para ler um número inteiro
%f	Usado para ler um número com casas decimais.
%c	Usado para ler um caractere
%s	Usado para ler uma sequencia de caracteres.

Entrada de Dados (Leitura de Dados)

Exemplos:

```
//Cálculo da área de um retângulo
#include <stdio.h>
main() {
    float base, altura, area;
    printf("Digite o valor da base: ");
    scanf("%f", &base);
    printf("Digite o valor da altura: ");
    scanf("%f", &altura);
    area = base * altura;
    printf("Area retangulo: %f", area);
}
```

Operadores Aritméticos

Símbolos utilizados para realizar operações aritméticas em expressões.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Bragança Paulista

Operadores Aritméticos

Os operadores aritméticos são símbolos que representam operações aritméticas, ou seja, as operações matemáticas básicas.

Operador	Operação matemática
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
--	**Decremento Unário**
++	**Incremento Unário**
%	Resto da Divisão Inteira

Operadores Aritméticos

Precedência de Operações

Os operadores tem uma dada preferência, que determina a ordem na qual as operações são executadas quando a **ausência de parênteses** causa ambiguidade na avaliação.

	Tipo	Símbolos
1	Operadores pós-fixados Operadores pré-fixados cast (conversão de tipo)	Exp++ exp— ++exp —-exp +exp –exp !exp (type) exp
2	Multiplicação/Divisão	* / %
3	Soma/Subtração	+ -

Operadores Aritméticos

Precedência de Operações

Os operadores tem uma dada preferência, que determina a ordem na qual as operações são executadas quando a **ausência de parênteses** causa ambiguidade na avaliação.

	Tipo	Símbolos
1	Operadores pós-fixados Operadores pré-fixados cast (conversão de tipo)	Exp++ exp— ++exp —-exp +exp –exp !exp (type) exp
2	Multiplicação/Divisão	* / %
3	Soma/Subtração	+ -

Dúvidas?

Materiais de Aula e Exemplos:

http://github.com/LuizAlbano