

Programação de Computadores IV

Formatação de entrada e saída

Slides cedidos por Bruno Augusto Dorta Marques

Roteiro da aula

- Saída padrão
 - Comando printf
- Formatação de entrada e saída
 - Sintaxe e Regras
- Entrada padrão
 - Comando scanf
 - Comando gets

Comando printf

- Imprimir na tela. Esse comando imprime uma sequência formatada na tela do computador. O comando possui a sintaxe

SINTAXE:

```
printf("formato", var1, var2, var3, ... varn)
```

Onde

“formato” é uma string com especificação do formato das variáveis a serem impressas.

var¹, var², var³, ... varⁿ é uma lista de variáveis a serem impressas.

Formatação de Entrada e Saída

- Tem como objetivo especificar a forma como os dados são lidos ou impressos
- A formatação pode ser utilizada em conjunto com os comandos `printf` (saída padrão) e `scanf` (entrada padrão)
- O formato é uma string contendo marcações especiais que iniciam com o símbolo %

Sintaxe da formatação

- Cada tipo de variável possui um identificador de formato:
- Tipo inteiro (int):
“%d”
- Tipo real (float, double)
“%f”
- Tipo caractere (char)
“%c”
- Tipo string (sequencia de char)
“%s”

Formatação números inteiros

- Utiliza-se o **%d**
- É possível especificar a quantidade mínima de dígitos que um número deve ser impresso.
- Para imprimir com no mínimo 4 dígitos, podemos especificar **“%4d”**
- Para imprimir com no mínimo 8 dígitos, podemos especificar **“%8d”**
- Também é possível imprimir com zeros a esquerda adicionado **0** após **%**

//Exemplo SINTAXE:

```
Int x = 0, y = 99, z = 87, w = -34344
```

```
printf(“%d”, x); //resultado na tela : 0
```

```
printf(“%04d %4d %d”, y, z, w); // resultado na tela: 0099 87 -34344
```

```
printf(“%d %d” , 6, 99) // resultado na tela: 6 99
```

Formatação números reais

- Utiliza-se o **%f**
- É possível especificar a quantidade mínima de dígitos você deseja que um número seja impresso. Além disso, é possível especificar a quantidade de casas decimais:
- Para imprimir com no mínimo 8 dígitos e 4 casas decimais, podemos especificar **“%8.4f”**

//Exemplo SINTAXE:

```
float x = 1.234, y = 29.342, z = 1.983746
```

```
printf(“%f”, x); // resultado na tela : 1.234000
```

```
printf(“%1.4f %4.2f %f”, x, y, z); // resultado na tela: 1.2340    29.34 1.983726
```

```
printf(“%f”, 2.876); // resultado na tela: 2.876
```

Formatação de strings

- Utiliza-se o %s

```
//Exemplo SINTAXE:
```

```
char palavra[100]
```

```
strcpy(palavra, "Olá, essa é uma string")
```

```
printf("%s", palavra);    //resultado na tela : Olá, isso é uma string
```

```
printf("%s %s", palavra, " e essa é outra string")
```

```
//resultado na tela: Olá, isso é uma string e essa é outra string
```

Formatação Extra

- Utiliza-se o `\n` para pular uma linha

```
//Exemplo SINTAXE:
```

```
char palavra[100]
```

```
strcpy(palavra, "Olá, essa é uma string")
```

```
printf("%s\n%s", palavra, " e essa é outra string")
```

```
//irá imprimir isso na tela:
```

```
Olá, isso é uma string
```

```
e essa é outra string
```

```
printf("%s\n\n%s", palavra, " e essa é outra string")
```

```
//irá imprimir isso na tela:
```

```
Olá, isso é uma string
```

```
e essa é outra string
```

Exemplo formatação

//Exemplo da impressão de uma tabela 2*2 com titulo:

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

```
void main(void){
    float nota1, nota2;
    char nome1[100], nome2[100];
    strcpy(nome1, "Joao");
    strcpy(nome2, "Maria");
    nota1 = 7.8;
    nota2 = 9.6;
```

```
    printf(" Nome | Nota \n%6s | %2.2f\n%6s | %2.2f", nome1, nota1, nome2, nota2);
}
```

Nome	Nota
Joao	7.80
Maria	9.60

Exercícios

Escreva um trecho de código que imprima todas as variáveis abaixo. Você deve imprimir cada variável em uma linha diferente.

```
int w, h, x;           // declaração de inteiros
float preco, z;        // declaração de número real
char palavra[100];     // declaração de sequência de caractere
char produto[100];

w = 3;
z = 2.85;
x = 5;
h = 8;
preco = 2.99;
strcpy(palavra, "programação");
strcpy(produto, "notebook");
```

Exercícios

Escreva um trecho de código que imprima todas as variáveis abaixo. Você deve imprimir cada variável em uma linha diferente.

```
int w, h, x;           // declaração de inteiros
float preco, z;        // declaração de número real
char palavra[100];     // declaração de sequência de caractere
char produto[100];

w = 3;
z = 2.85;
x = 5;
h = 8;
preco = 2.99;
strcpy(palavra, "programação");
strcpy(produto, "notebook");

printf("%d\n%d\n%d\n%f\n%f\n%s\n%s\n", w, h, x, z, preco, palavra, produto);
```

Exercícios

Supondo que deseja-se armazenar as variáveis: Dia, mês, ano, e o nome do usuário.

Escreva um programa que imprima a seguinte mensagem na tela:

```
Ola, bem vindo Bruno. Data de hoje : |13/ 9/2016|
```

Exercícios

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

void main(void){

    int dia, mes, ano;
    char nome[100];
    strcpy(nome, "Bruno");

    dia = 13;
    mes = 9;
    ano = 2016;

    printf("_____\\nOla, bem
vindo %s. Data de hoje : |%2d/%2d/%4d|\\n", nome, dia, mes, ano);
}
```

Comando scanf

- A entrada padrão é o teclado. Esse comando recebe dados pelo teclado do computador e os armazena em uma variável. O comando possui a sintaxe

SINTAXE:

```
scanf("formato", &var1, &var2, &var3, ... &varn)
```

Onde

“formato” é uma string com especificação do formato das variáveis a serem impressas.

&var¹, &var², &var³, ... &varⁿ é uma lista de variáveis a serem impressas.

Atenção... É necessário inserir “&” pois a função scanf recebe o endereço das variáveis.

Exemplo de uso - scanf

```
int w, h, x;           // declaração de inteiros
float preco, z;        //declaração de número real
char palavra[100];     //declaração de sequência de caractere
char produto[100];

scanf("%d", &w);        //recebe um inteiro do teclado e armazena o valor em w
scanf("%d,%d", &h,      //recebe dois inteiros do teclado e armazena o valor em h e x
    &x);
scanf("%f, %f, %s, %s", &preco, &z, &palavra, &produto );
//o comando acima recebe o valor de dois floats e duas strings.

printf("%d\n%d\n%d\n%f\n%f\n%s\n%s\n", w, h, x, z, palavra, produto);
```

Comando gets

- O comando scanf lê a entrada até que encontre uma quebra de linha ou espaço em branco.

Para ler uma string que contenha espaço em branco, é necessário utilizar o comando **gets** :

```
#include<stdio.h>
int main (void){
    char frase[100];           //declaração de sequência de caractere

    scanf("%s", &frase);       //recebe uma frase até encontrar um espaço em branco
    printf("%s", frase);       //imprime o valor armazenado em frase

    gets(frase);               //recebe uma String e armazena na variável frase
    printf("%s", frase);       //imprime o valor armazenado em frase
    return 0;
}
```