Programação de Computadores I

Aula 06

Programação: Entrada e Saída de Dados

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2011-1

Saída de dados I

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando a função *printf* da biblioteca **stdio.h**.
- O primeiro argumeneto da função `printf` é uma string usada para formatar a saída de dados. Nesta string utilizamos especificadores de formatação para inserir os valores que desejamos na saída. Um especificador de formatador começa com o caracter `%`. Para incluir o próprio `%` na saída, escreve-se este caracter duas vezes: `%%`.
- A cada especificador de formatação corresponde um argumento adicional na chamada de função, cujo valor será inserido no lugar do formatador para produzir a saída desejada.

Saída de dados II

- Ex.
  `printf ("A variável %s contém o valor %d", "a", a);`
- Imprime: A variável *a* contém o valor 10
- Nesse caso, `%s` deve ser substituído por uma variável ou constante do tipo *string* enquanto `%d` deve ser substituído por uma variável do tipo inteiro.

Formatos inteiro I

- `%d`: escreve um número inteiro na tela em notação decimal.

  Ex: `printf("%d", 10);`

  Imprime 10

Formatos inteiro II

- `%< número >d`: escreve um inteiro na tela, preenchendo com espaços a esquerda para que ele ocupe pelo menos `< número >` casas na tela.

  Ex: `printf("%4d", 10);`

  imprime ␣␣10

- `%0< número >d`: escreve um inteiro na tela, preenchendo com zeros a esquerda para que ele ocupe pelo menos `< número >` casas na tela.

  Ex: `printf("%04d", 10);`

  imprime 0010

Formatos inteiro III

- A letra *d* pode ser substituída pelas letras *u* e *l*, ou as duas, quando desejamos escrever variáveis do tipo *unsigned* ou *long*, respectivamente.

  Ex: `printf("%d", 4000000000);`

  escreve -294967296 na tela, enquanto que

  `printf("%u", 4000000000);`

  escreve 4000000000

Formatos ponto flutuante I

- `%f`: escreve um valor do tipo **`double`** na tela, sem formatação

  Ex: `printf("%f", 10.0);`

  imprime 10.000000

Formatos ponto flutuante II

- `%e`: escreve um **`double`** na tela, em notação científica
  - Ex: `printf("%e", 10.02545);`
  - imprime $1.002545e+01$

Formatos ponto flutuante III

- `%< tamanho >.< decimais >f`: escreve um **`double`** na tela, com tamanho `< tamanho >` e `< decimais >` casas decimais.
- O ponto utilizados para separar a parte inteira da decimal, também conta no tamanho

  Ex: `printf("%6.2f", 10.0);`

  ␣10.00

Formato caracter

- `%c`: escreve uma letra.

  Ex. `printf("%c", 'A');`
  imprime a letra A

  Note que `printf("%c", 65);` também imprime a letra A.

Formato *string* I

- `%s`: escreve um *string*

  Ex. `printf ("%s", "Meu primeiro programa");`

  imprime Meu primeiro programa

Exercícios para ser resolvidos em aula I

1. Faça um algoritmo que lê o número de um funcionário, seu número de horas trabalhadas e o valor que recebe por hora. O algoritmo deve calcular e mostrar o salário deste funcionário.
2. Faça um algoritmo para ler dois inteiros (variáveis A e B) e efetuar as operações de adição, subtração, multiplicação e divisão de A por B apresentando ao final os quatro resultados obtidos.
3. Identifique e corrija os erros em cada uma das instruções a seguir. Pode haver mais de um erro por instrução. Escrever as mesmas mensagens usando `cout`

   3.1 `printf("o produto de %d e %d eh %d\n", x, y);`

   3.2 `primeiroNumero + segundoNumero = somaTotal`

   3.3 `print("A soma eh %d", x+y);`

   3.4 `Printf("o valor fornecido eh %d:", &valor);`

Exercícios propostos I

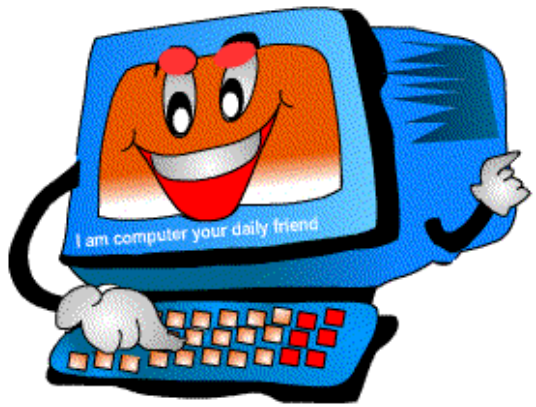
- Faça um algoritmo que lê o código da peça 1, a quantidade vendida de peças 1, o valor unitário da peça 1, o código da peça 2, a quantidade vendida de peças 2, o valor unitário da peça 2 e a porcentagem do IPI a ser acrescentada. O algoritmo deve calcular o valor total a ser pago.
- Faça um algoritmo que calcule e mostre a área de um trapézio. Sabe-se que a área é definida por:

$$A = ((basemaior + basemenor) \times altura)/2.$$

- Uma loja de animais precisa de um algoritmo para calcular os custos de criação de coelhos. O custo é calculado com a fórmula $C = ((NCOELHOS * 0.70)/18) + 10$. O algoritmo tem como entrada o número de coelhos, devendo fornecer, como saída, o custo.

A função `scanf` I

- realiza a leitura de um texto a partir do teclado
- parâmetros:
  - uma *string*, indicando os tipos das variáveis que serão lidas e o formato dessa leitura.
  - uma lista de variáveis
- aguarda que o usuário digite um valor e atribui o valor digitado à variável



A função `scanf` II

```c
#include <stdio.h>
int main(void)
{
    int n;
    printf("Digite um número: ");
    scanf("%d", &n);
    printf("O valor digitado foi %d\n", n);
    return 0;
}
```

A função `scanf` III

O programa acima é composto de quatro passos:

- Cria uma variável *n*;
- Escreve na tela *Digite um número:*
- Lê o valor do número digitado
- Imprime o valor do número digitado

A função `scanf` IV

Leitura de várias variáveis

```c
#include <stdio.h>
int main(void)
{
    int m, n, o;
    printf("Digite três números: ");
    scanf("%d%d%d",&m, &n, &o);
    printf("O valores digitados foram %d %d %d\n", m, n, o);
    return 0;
}
```

O endereço de uma variável I

- Toda variável tem um **endereço de memória** associada a ela.
- Esse endereço é o **local** onde essa variável é armanenada no sistema (como se fosse o endereço de uma casa, o local onde as pessoas são “armazenadas”). Ou ainda como se fosse o número de uma caixa postal, onde correspondências são armazenadas.

Endereço	Conteúdo	Variável
⋮	⋮	⋮
1001	???	
1002	2450	i
1003	???	
1004	225.345	f
1005	11331	j

O endereço de uma variável II

- Normalmente, o endereço das variáveis não são conhecidos quando o programa é escrito.
- O endereço é dependente do sistema computacional e também da implementação do compilador C que está sendo usado.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

O operador de endereço & de C I

- O operador & retorna o endereço de uma determinada variável
- Imprime o endereço (no formato hexadecimal) da variável `valor`:

```
printf("%x", &valor);
```

O operador de endereço `&` de C II

- É necessário usar o operador `&` na função `scanf`, pois esse operador indica que o valor digitado pelo usuário deve ser armazenado no endereço de memória referente à variável.
- Esquecer de colocar o operador de endereço (`&`) é um erro muito comum que pode ocasionar sérios problemas na execução do programa.

O operador de endereço & de C III

O programa abaixo imprime o valor e o endereço da variável:

```
#include <stdio.h>
int main(void)
{
    int n = 8;
    printf("valor %d, endereço %x\n", n, &n);
    return 0;
}
```

Formatos de leitura com `scanf`

- Os formatos de leitura são semelhantes aos formatos de escrita utilizados pelo `printf`.
- As tabelas seguintes mostram alguns formatos possíveis de leitura, começando com números.

Formatos de leitura: números I

Código	Função
`%d`	Lê um número inteiro em notação decimal
`%u`	Lê um número inteiro decimal sem sinal
`%l`	Lê um inteiro longo
`%lu`	Lê um inteiro longo sem sinal
`%hd`	Lê um *short*
`%hu`	Lê um *short* sem sinal
`%f`	Lê um *float*
`%lf`	Lê um *double*
`%Lf`	Lê um *long double*

Formatos de leitura: números II

Exemplo: escreva um programa que permita inserir a idade e o peso de um indivíduo.

```
#include <stdio.h>
int main(void)
{
    int idade;
    double peso;
    printf("Digite a idade: ");
    scanf("%d", &idade);
    printf("Digite o peso: ");
    scanf("%lf", &peso);
    printf("O indivíduo tem %d anos e pesa %f kg.\n",
           idade, peso);
    return 0;
}
```

Formatos de leitura: tamanho máximo do campo

- O tamanho máximo de um campo pode ser especificado em um formato imediatamente após o caracter `%`.
- A leitura dos caracteres termina quando o máximo é atingido ou quando um caracter inválidd é encontrado.
- Exemplo:

```c
#include <stdio.h>
int main(void)
{
    int idade;
    double peso;
    printf("Digite a idade: ");
    scanf("%2d", &idade);
    printf("Digite o peso: ");
    scanf("%lf", &peso);
    printf("Idade: %d, peso: %f\n", idade, peso);
    return 0;
}
```

Se for digitado `18620`, a idade será 18 e o peso será 620.

Formatos de leitura: brancos no começo da entrada

- Geralmente se os primeiros caracteres da entrada forem brancos (espaço, tabulação, mudança de linha, etc.), eles são pulados.
- Uma sequência de caracteres brancos na string de formatação faz com que todos os brancos no começo da entrada sejam pulados. Isto pode ser último em algumas situações que veremos mais adiante.
- Exemplo:

```
#include <stdio.h>
int main(void)
{
    int idade;
    double peso;
    printf("Digite a idade e o peso: ");
    scanf("%d   %lf", &idade, &peso);
    printf("Idade: %d, peso: %f\n", idade, peso);
    return 0;
}
```

Formatos de leitura: ignorar atribuição

- Pode-se ignorar uma entrada colocando-se um `*` logo depois do `%` na especificação de formato.
- Um valor vai ser lido de acordo com a especificação usada.
- Porém o valor lido não será atribuído a nenhuma variável.
- Não se coloca nenhum endereço de variável correspondente a este formato como argumento para `scanf`.
- Exemplo:

```
#include <stdio.h>
int main(void)
{
    double peso;
    printf("Digite o peso e a altura: ");
    scanf("%lf%*lf", &peso);
    printf("Peso: %f\n", peso);
    printf("A altura foi ignorada\n");
    return 0;
}
```

Formatos de leitura: caracter

Código	Função
`%c`	▶ lê um único caracter ▶ não pula brancos no início ▶ para pular brancos no início, use um espaço na string de formatação

Exemplo:

```
#include <stdio.h>
int main(void)
{
    char car;
    printf("Digite um caracter: ");
    scanf("%c", &car);
    printf("O caracter digitado foi: %c\n", car);
    return 0;
}
```

Formatos de leitura: string I

Código	Função
%s	▶ lê uma sequência de caracteres não brancos (*string*) ▶ entrada pára no primeiro caracter branco ▶ deve haver espaço suficiente para todos os caracteres lidos mais o caracter nulo que finaliza a *string*; se não houver, ocorre erro lógico.

Formatos de leitura: string II

Exemplo:

```c
#include <stdio.h>
int main(void)
{
    char nome[20];
    printf("Digite o nome: ");
    scanf("%s", nome);
    printf("Olá %s\n", nome);
    return 0;
}
```

- Se forem digitado mais de 19 caracteres ocorrerá um erro, pois o número de caracteres armazenados é maior do que a quantidade de espaço reservada.
- Não é possível ler um nome formado por mais de uma palavra, como por exemplo *Maria Antonieta*.

Formatos de leitura: string III

Código	Função
`%[`*conjunto de caracteres*`]`	▶ lê uma sequência não vazia de caracteres especificada pelo conjunto ▶ entrada pára no primeiro caracter que não pertencer ao conjunto ▶ não pula brancos no início ▶ deve haver espaço suficiente para todos os caracteres lidos mais o caracter nulo que finaliza a *string*; se não houver, ocorre erro lógico.

Formatos de leitura: string IV

Exemplos:

Código	Função
`%[aeiou]`	Lê uma sequência de vogais
`%[0-9]`	Lê uma sequência de dígitos decimais
`%[0-9A-Fa-f]`	Lê uma sequência de dígitos hexadecimais
`%[a-zA-Z0-9]`	Lê uma sequência de caracteres alfanuméricos
`%[^0-9]`	Lê uma sequência de caracteres que não são dígitos decimais
`%[^\n]`	Lê uma sequência de caracteres quaisquer, exceto a mudança de linha

Formatos de leitura: string V

Exemplo:

```c
#include <stdio.h>
int main(void)
{
    char nome[20];
    char sexo;
    printf("Digite o nome completo: ");
    scanf(" %19[^\n]%*[^\n]", nome);
    printf("Sexo (m/f): ");
    scanf(" %c", &sexo);
    printf("Olá %s; sexo: %c\n", nome, sexo);
    return 0;
}
```

- Se forem digitado mais de 19 caracteres não ocorrerá um erro.
- Quaisquer caracteres excedentes serão descartados.
- A mudança de linha também não será descartada no primeiro `scanf`.

A função `gets` I

- A função `scanf` não aceita espaços em branco com o `%s`. A função `gets` aceita todo tipo de caractere, incluindo o espaço em branco.
- Protótipo:

```
char *gets(char *str);
```

- É necessário que haja espaço suficiente na string para armazenar os caracteres que serão lidos e o caracter nulo que será colocado no final string. Caso não haja o programa não funcionará corretamente.

A função gets II

Exemplo:

```c
#include <stdio.h>
int main(void)
{
    char nome[50];
    int idade;
    printf("Inserir o nome completo: ");
    gets(nome);
    printf("Inserir idade: ");
    scanf("%d", &idade);
    printf("%s tem %d anos", nome, idade);
    return 0;
}
```

- Não é recomendável o uso da função gets
- Não existe forma de controlar a quantidade de caracteres a serem inseridos
- Utilizar uma função mais segura como fgets

A função `fgets` I

- A função `fgets` permite especificar a quantidade de caracteres que podem ser armazenadas na variável onde será armazenada a string.
- Protótipo:

```
char *fgets(char *str, int size, FILE *fluxo);
```

- O caracter de mudança de linha `'\n'` encontrado no final da entrada é inserido no final da string, seguido do caracter nulo.
- Porém qualquer caracter excedente continuará disponível na entrada e será consumido na próxima operação de leitura.

A função `fgets` II

Exemplo:

```
#include <stdio.h>
int main(void)
{
    char nome[50];
    int idade;
    printf("Digite o nome completo: ");
    fgets(nome, 50, stdin);
    printf("Digite a idade: ");
    scanf("%d", idade);
    printf("%s tem %d anos\n", %s, %d);
    return 0;
}
```

Exercícios em Aula

1. Escreva um programa que, dado um número de segundos, converte-o para dias, horas, minutos e segundos. Por exemplo, 7322 segundos correspondem a 0 dias, 2 horas, 2 minutos e 2 segundos.
2. Escreva um programa que determina quanto tempo (t) um corpo em repouso ($v_0 = 0$) leva para atingir o solo ($h = 0$) a partir de uma altura h_0, informada pelo usuário. Considere $g = -9,8 m/s^2$ e que altura da queda livre é determinada pela fórmula $h = h_0 + v_0 t + \frac{1}{2} g t^2$.
3. Faça um programa que leia um número e exiba seu sucessor.
4. Faça um progrma para ler dois números inteiros, x e y, e imprimir o quociente e o resto da divisão inteira entre eles.

Exercícios em Propostos

1. Escreva um programa para ler 3 números reais do teclado e verificar se o primeiro é maior que a soma dos outros dois.
2. Faça um programa que gere o preço de um carro ao consumidor e os valores pagos pelo imposto e pelo lucro do distribuidor, sabendo o custo de fábrica do carro e que são pagos:
   - de imposto: 45% sobre o custo do carro;
   - de lucro do distribuidor: 12% sobre o custo do carro.
3. Escrever um programa para ler 4 números inteiros e calcular a soma dos que forem par.
4. Fazer um programa que informe se um dado ano é ou não bissexto. Obs: um ano é bissexto se ele for divisível por 400 ou se ele for divisível por 4 e não por 100.

FIM

Créditos:
Baseado no material preparado pelo
Prof. Guillermo Cámara-Chávez.