Universidade Católica de Brasília

Algoritmo e Programação Estruturada

Aula 2

Prof. Osmam Brás de Souto

Funções de Entrada e Saída

FUNÇÃO *printf*()

- Uma das funções de saída que pode ser usada em C
- A <expressão_controle> pode conter caracteres que serão exibidos na tela e códigos de formatação dos argumentos
- A <lista_de_argumentos> é formada pelos argumentos que serão usados pela função (constantes, variáveis,expressões,etc.)
- Faz parte da biblioteca padrão da linguagem C e tem seu protótipo definido em (*stdio.h*) e pode receber um número variável de argumentos (separados por ‘**,**’)

printf(“<expressão_controle>”,<lista_de_argumentos>);

Exemplo: printf(“Elaborando um programa”);

printf(“Esta é a letra %c”, ‘C’);

printf(“Escreva o valor %d \n %s”,2, “por extenso”);

Funções de Entrada e Saída

FUNÇÃO *printf*()

Alguns caracteres não podem ser obtidos diretamente do teclado (como mudança de linha), por isso tem a sua representação feita por códigos especiais usando um controlador (‘\’) com a combinação de outro caractere.

Código	Significado
\n	nova linha
\r	retorno do cursor
\t	tab
\b	retrocesso
\”	aspas
\\	barra
\f	salta página de formulário
\0	nulo
\xdd	caractere gráfico (este símbolo escreve códigos acima de 127 decimal) \xdd onde dd é hexad.

Código	Formato
%c	caractere simples
%d	decimal
%e	notação científica
%f	ponto flutuante
%g	%e ou %f (o mais curto)
%o	octal
%s	cadeia de caracteres
%u	decimal sem sinal
%x	hexadecimal
%ld	decimal longo
%lf	ponto flutuante longo

Funções de Entrada e Saída

FUNÇÃO *printf*()

- Para a impressão do caractere % são usados dois % → " %% "
- É possível estabelecer o tamanho mínimo de saída em printf
- A formatação com '-' à frente do formato alinha a saída a esquerda, enquanto que a direita consegue-se pela especificação coerente dos tamanhos
- A saída formatada pode ser completada com zeros a esquerda

Exemplos:

1) printf("Reajuste = %d %%",valor); saída Reajuste = 5 %

2) printf("Tem %4d alunos", 50); saída Tem 50 alunos

2) printf("Gasolina= %3.2f ltr.", 1.629); saída Gasolina=1.63 ltr.

3) printf("%5.2f %5.2f ",8.0, 15.32); saída 8.00 15.32

3) printf("%5.2f %5.2f ",15.8, 0.7); saída 15.80 0.70

3) printf("%-5.2f %-5.2f ",8.0, 15.32); saída 8.00 15.32

3) printf("%-5.2f %-5.2f ",15.8, 0.7); saída 15.80 0.70

4) printf("%06d", 25); saída 000025

Funções de Entrada e Saída

FUNÇÃO *scanf*()

- Permite ler dados formatados da entrada padrão (teclado)
- A <expressão_controle> deve conter os códigos compatíveis com a formatação dos argumentos (similar ao *printf*)
- A <lista_de_argumentos> nesta função deve conter os endereços das variáveis (usar operador de endereço (&) em C)
- O símbolo %* indica que será lido um valor do tipo especificado, mas não será atribuído a nenhuma variável (não devendo ter parâmetros na lista de argumentos)
- Faz parte da biblioteca padrão da linguagem C, necessita dos protótipos definidos em (*stdio.h*) e pode receber um número variável de argumentos (separados por '**,**')

scanf("<expressão_controle>",<lista_de_argumentos>);

Exemplo: scanf("%f ", &valor);

endereços

Funções de Entrada e Saída

FUNÇÃO *scanf*()

O endereço de uma variável é o local onde ela localiza-se na memória. O endereço representa o primeiro byte ocupado por ela

Código	Formatação da Função
%c	leia um único caractere
%d	leia um inteiro decimal
%e	leia um número em notação científica
%f	leia um número em ponto flutuante
%o	leia um inteiro octal
%s	leia uma série de caracteres
%u	leia um decimal sem sinal
%x	leia um número hexadecimal
%l	leia um inteiro longo
%lf	leia um double

Exemplo:

```
int main(void)
{
int num;
num = 5;
printf(“Valor= %d\n”, num);
printf(“Endereço=%u,&num );
}
```

Possível saída em tela

```
Valor=5
Endereço=2293620
```

Funções de Entrada e Saída

FUNÇÃO *clrscr*() (include <conio.c>)

Esta função efetua a limpeza de toda a tela (ou janela 24x80). Esta limpeza consiste em apagar todos os símbolos (ou caracteres) que estavam sendo apresentados em instruções anteriores.

Com a limpeza de toda a janela, o cursor fica posicionado na primeira posição da tela, ou seja, no canto superior esquerdo.

Exemplo:

```
#include <iostream>
int main() {
    float valor;
    printf("Qual o valor agregado: \n");
    scanf("%f",&valor);
    system("cls");
    printf("O resultado %f", valor);

}
```

system("cls");

Funções de Entrada e Saída

➔ FUNÇÃO *getch*() (include <conio.c>)

Esta função lê um caractere do teclado, no instante em que foi pressionado, e não apresenta o caractere na tela

➔ Exemplo:

```
int main() {
    float valor;
    printf("Qual o valor agregado: \n");
    scanf("%f",&valor);
    system("pause");
    printf("O resultado %f", valor);

}
```

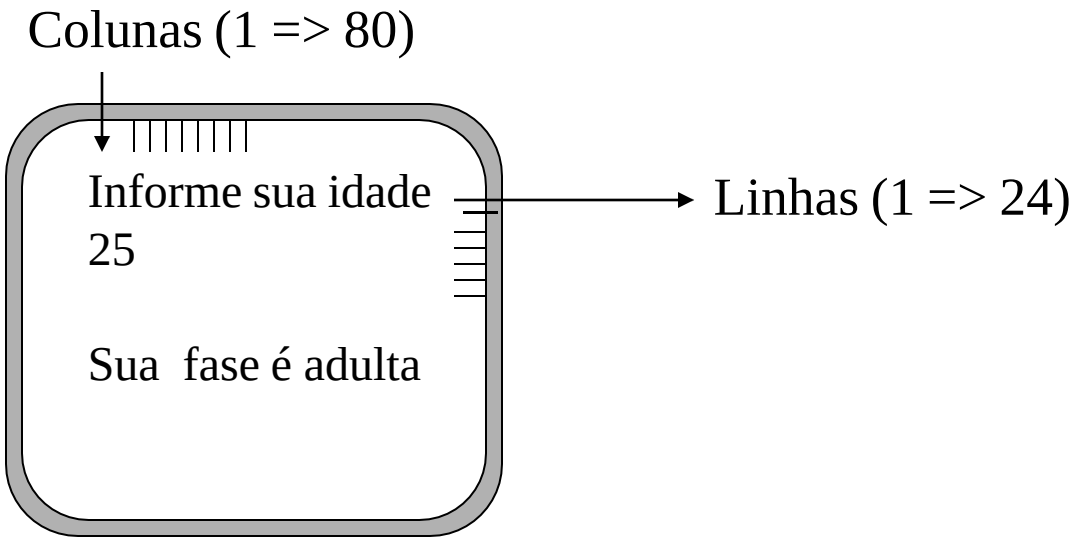
system("pause");

Funções de Entrada e Saída

A execução de um programa elaborado no compilador C abre uma nova janela que possui um tamanho padrão. Este tamanho permanece o mesmo do ambiente DOS, mas agora fazendo a rolagem da janela, quando ela é necessária (extrapola o tamanho padrão).

Para aproveitar melhor toda janela, sem usar a rolagem, trabalha-se com uma dimensão de 24 linhas, por 80 colunas (tamanho padrão do DOS)..

Obs.: A função **clrscr()** limpa só o que estiver contido na tela padrão(24x80)



Funções de Entrada e Saída

Conhecendo as medidas da janela de saída pode-se trabalhar melhor a execução de um programa.

Com este intuito, é usada a função *gotoxy* que permite o posicionamento correto na janela de execução.

FUNÇÃO *gotoxy*() (include <conio.c>)

Esta função permite o posicionamento exato do cursor, possibilitando uma apresentação mais amigável na execução de qualquer programa em C.

gotoxy(<coluna> , <linha>);

Exemplo:

```
int main() {
    float valor;
    printf("Qual o valor agregado: \n");
    scanf("%f",&valor);
    gotoxy(24,11);
    printf("O resultado %f", valor);
}
```

Universidade Católica de Brasília

Referência de Criação e Apoio ao Estudo

Material para Consulta e Apoio ao Conteúdo

- FARRER, H. et all, Algoritmos Estruturados, Editora LTC, 3ª . edição, 1999. - livro
  - Capítulo 0

- MANZANO, J. e Oliveira, J., Algoritmos, Lógica para desenvolvimento de programação, Editora Ética, 1996.
  - Capítulo 1