

Cap. IV – COMANDOS DE ENTRADA E SAÍDA

4.1 – Equipamentos de Entrada e Saída

- Comunicação com o exterior:
 - Recepção de informações: equipamentos de entrada
 - Emissão de informações: equipamentos de saída
- Equipamentos de entrada:
 - teclado (com *eco* no vídeo)
 - mouse
 - discos (winchester e disquetes)
 - joystick
 - microfone
 - leitora ótica
 - célula foto-sensora
 - cd-player
 - câmera
 - fita
 - leitora de cartões
- Equipamentos de saída:
 - vídeo (texto ou gráfico)
 - impressoras diversas
 - auto-falantes
 - discos
 - plotters
 - fita

Neste capítulo:
entrada pelo teclado e
por arquivos em disco;
saída no vídeo e em
arquivos em disco.

4.2 – Saída no Vídeo-Texto

4.2.1 – Função *putchar* ()

- Escreve no vídeo um caractere fornecido como argumento
- **Exemplo 4.1:** os comandos

```
c = 'b';  
putchar (c + 'A' - 'a');
```

escrevem 'B' no vídeo

4.2.2 – Função *printf* ()

a) Forma:

```
printf ( cadeia de controle , outros argumentos );
```

- **Exemplo 4.2:** seja o comando

```
printf ( “A loja vende %d %s por R$%f” , 5,  
        “compact-disks” , 60.00);
```

cadeia de controle : “A loja vende %d %s por R\$%f”

outros argumentos : 5, “compact-disks” , 60.00

Saída no vídeo:

A loja vende 5 compact-disks por R\$60.000000

- São impressos todos os caracteres da cadeia de controle que estão fora de um *formato*;
 - O início de um formato é o caractere ‘%’ .

- Tabela dos formatos usuais:

Formato	Tipo da expressão
%c	Caractere
%d , %i	Inteiro decimal com 2 bytes
%ld	Inteiro decimal com 4 bytes
%u	Inteiro decimal, sem sinal
%o	Inteiro octal, sem sinal
%x , %X	Inteiro hexadecimal, sem sinal
%f	Real com 6 casas decimais
%g	Real com no máximo 6 dígitos significativos
%e , %E	Real em notação exponencial
%s	Cadeia de caracteres

b) Especificação do campo para impressão

- **Exemplo 4.3:** seja o seguinte comando:

```
printf ("***%c**%3c**%-3c***%s**%8s**%-8s***",
        'A', 'A', 'A', "ABCDE", "ABCDE", "ABCDE");
```

o resultado exibido será:

***A** A**A ***ABCDE** ABCDE**ABCDE ***

Espaços em branco

- **Exemplo 4.4:** seja o seguinte comando:

```
printf ("***%c***%d***%c***%d***%4d***%-4d***",
        'B', 'B', 69, 69, 35, 35);
```

o resultado exibido será:

```
***B***66***E***69***__35***35__***
```

- **Obs:** Um caractere no formato %d exibe seu código ASCII e um inteiro no formato %c exibe o caractere cujo código ASCII tem o seu valor.

c) Especificação da precisão de números reais

- No formato %f, usa-se dois números separados por um ponto; o segundo número é o número de casas depois do ponto decimal.

- **Exemplo 4.5:** seja o comando:

```
printf ("x =%20.10f", 1234.12344347263849);
```

resultado: x_ = 1234.1234434726

10 casas

20 casas

- Nos formatos %d e %s, o segundo número tem interpretação diferente.
- Exemplo 4.6:** sejam os seguintes comandos:

1) printf ("x = %10.7d", 1234);

resultado: x_ = $\underbrace{\quad\quad\quad\boxed{0001234}\quad\quad\quad}_{10 \text{ casas}}$
7 casas

2) printf ("cad =%10.6s", "ABCDEFGH")

resultado: cad_ = $\underbrace{\quad\quad\quad\boxed{ABCDEF}\quad\quad\quad}_{10 \text{ casas}}$
6 casas

d) Variação do campo para impressão e da precisão

- O campo para a impressão e a precisão podem variar de acordo com o valor de expressões e de variáveis do programa.

- Seja o comando

printf ("valor =%*.*f", *expr1*, *expr2*, *expr3*);

O valor de *expr3* será impresso num espaço de *expr1* casas, com *expr2* dígitos depois do ponto decimal.

- Exemplo 4.7:** seja o programa:

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

```
void main () {  
    double x; int m, n;  
    x = 337.3777337337;  
    for (n = 1; n <= 4; n++) {  
        m = 8 + n;  
        printf ("### %*.*f ###\n", m, n, x);  
    }  
}
```

resultado:

```
###____337.4###  
###____337.38###  
###____337.378###  
###____337.3777###
```

Obs: houve arredondamento

- **Exemplo 4.8:** curva de uma função

O programa a seguir traça uma curva para a função

$$y = x^2 - 2x - 3$$

no intervalo [-2, +4] para a variável x; pressupõe-se o eixo das ordenadas horizontal e o eixo das abscissas vertical, passando pela coluna 10 da janela de execução.

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

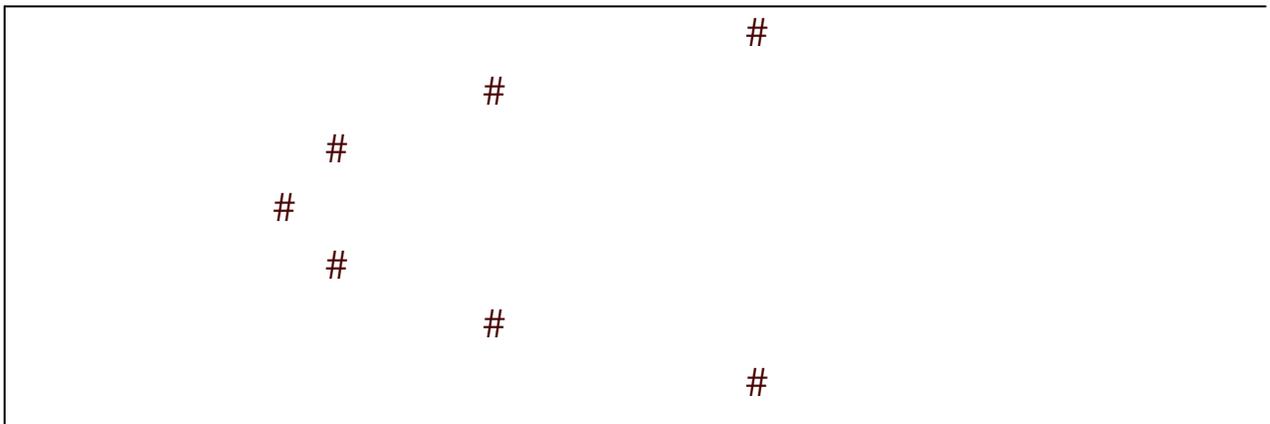
```

void main ()
{
    int x;
    for (x = -2; x <= 4; x++)
        printf ("%*c\n", 10 + x*x - 2*x - 3, '#');
}

```

Resultado:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21



- **Exemplo 4.9:** melhoramentos no gráfico anterior

O programa a seguir acrescenta ao gráfico anterior uma régua e os eixos das coordenadas da função

$$y = x^2 - 2x - 3$$

O intervalo de variação de x é $[-5, +7]$

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

```
void main ()
```

```
{
```

```
    int x, j, func;
```

```
/* Tracado da regua */
```

```
    printf ("----|----|----|----|----|----|----|----|\n");
```

```
/* Calculo da funcao no intervalo [-5, +7] */
```

```
    for (x=-5;x<=7;x++) {  
        func = 5 + x*x - 2*x - 3;
```

```
/* Se x!=0, marcar o ponto da curva e um ponto do eixo das abscissas */
```

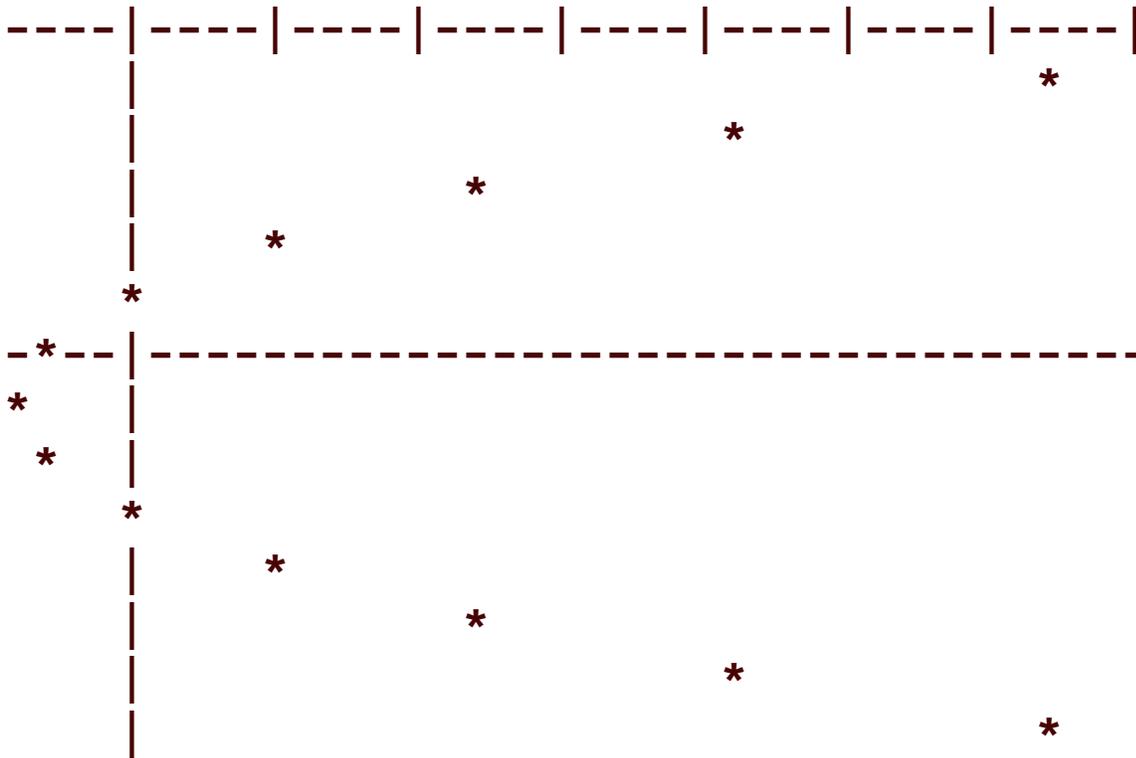
```
        if (x != 0) {  
            if (func < 5) printf ("%*c%*c", func, '*', 5-func, '|');  
            else if (func == 5) printf ("%*c", func, '*');  
            else printf ("%*c%*c", 5, '|', func-5, '*');  
        }  
    }
```

```
/* Se x=0, tracar tambem o eixo das ordenadas */
```

```
        else  
            for (j=1; j<=40; j++)  
                if (j == func) printf ("%c", '*');  
                else if (j == 5) printf ("%c", '|');  
                else printf ("%c", '-');  
            printf ("\n");  
        }  
    }
```

```
}
```

Resultado:



4.3 – Entrada pelo teclado

4.3.1 – Funções *getch* () e *getche* ()

- As funções *getch* () e *getche* () aguardam o operador digitar uma tecla (do teclado) e retornam o caractere correspondente a essa tecla.
- A diferença entre elas é que a função *getche* () dá “eco” no vídeo e a função *getch* () não dá.
- Ambas estão no arquivo **conio.h** da biblioteca de C.
- **Exemplo 4.10:** seja o seguinte programa:

```

#include <stdio.h>
#include <conio.h>

void main ()
{
    char c;
    printf ("Digite uma tecla:"); c = getch ();
    printf ("\nTecla digitada:%c", c);
    printf ("\nDigite uma tecla:"); c = getch ();
    printf ("\nTecla digitada:%c", c);
}

```

Se as teclas digitadas forem ‘b’ e ‘v’, o resultado será:

```

    Digite uma tecla:
    Tecla digitada:b
    Digite uma tecla:v
    Tecla digitada:v

```

4.3.2 – Função *getchar* ()

- As funções *getchar* () e *scanf* () não fazem leitura diretamente do teclado; elas lêem de uma entidade chamada “**buffer do teclado**”.

- Se o referido buffer estiver vazio, elas esperam o operador digitar um conjunto de caracteres encerrado por um “enter”.
- Todos os caracteres digitados, inclusive o “enter”, ficam guardados no buffer de entrada.
- Uma vez cheio o buffer, então as funções *getchar* () e *scanf* () fazem a leitura.
- Uma leitura pode não consumir todo o buffer; se o buffer não estiver vazio no início de uma leitura, tais funções fazem a leitura sem aguardar digitação por parte do operador.
- **Exemplo 4.11:** seja o seguinte programa:

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

```
void main ()
{
    char c;
    printf ("Escreva uma sentença encerrada por \n");
    while ((c = getchar()) != '\n')
        if (islower(c) != 0) putchar (c + 'A' - 'a');
        else putchar (c);
}
```

Só na primeira vez que a função *getchar* () for executada, ela esperará a digitação que preencherá o buffer. Nas outras vezes, o buffer não estará vazio.

Se o buffer for preenchido com:

abCX98;ynnngVF.*@ef “enter”

o resultado será:

ABCX98;YNNGGVF.*@EF

ou seja, as letras minúsculas digitadas foram trocadas pelas correspondentes maiúsculas.

Se o buffer for preenchido apenas com um “enter”, o programa se encerrará sem exibir nada no vídeo.

4.3.3 – Função *scanf* ()

- Forma:

scanf (*cadeia de controle, outros argumentos*);

- **Exemplo 4.12:** seja o seguinte trecho de programa:

```
char a, b; int n; double x;  
scanf (“%c%c%d%lf”, &a, &b, &n, &x);
```

cadeia de controle: “%c%c%d%lf”
outros argumentos: &a, &b, &n, &x

Os outros argumentos devem ser endereços

- Tabela de formatos usuais:

Formato	Tipo da variável
%c	Caractere
%d, %i, %u, %o, %x, %X	Inteiro de 2 bytes
%ld	Inteiro de 4 bytes
%f, %g, %G, %e, %E	Real de precisão simples
%lf, %le	Real de dupla precisão
%s	Cadeia de caracteres

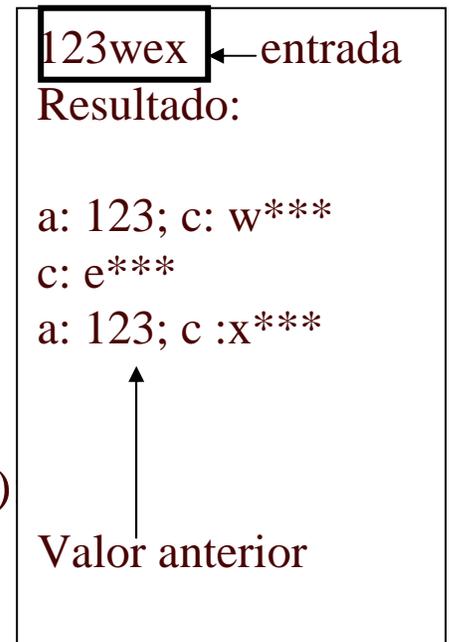
- A leitura de um caractere incompatível com o formato de leitura de um argumento:
 - interrompe a leitura do argumento;
 - mantém o caractere no buffer de leitura;
 - dá ao argumento o valor formado até então.

- **Exemplo 4.13:** sejam os comandos:

```

char c; long a;
scanf ("%ld%c", &a, &c);
printf ("a: %ld; c: %c***\n", a, c);
scanf ("%c", &c);
printf ("c: %c***\n", c);
scanf ("%ld", &a); scanf ("%c", &c)
printf ("a: %ld; c: %c***\n", a, c);

```



4.4 – Entrada e Saída por Arquivos em Disco

- Todo arquivo do diretório do usuário precisa ser associado a uma variável do tipo **FILE** do programa, para ser acessado por esse programa.
- Comandos para essa associação:

```
FILE *fl;  
fl = fopen ( nome_do_arquivo, modo);
```

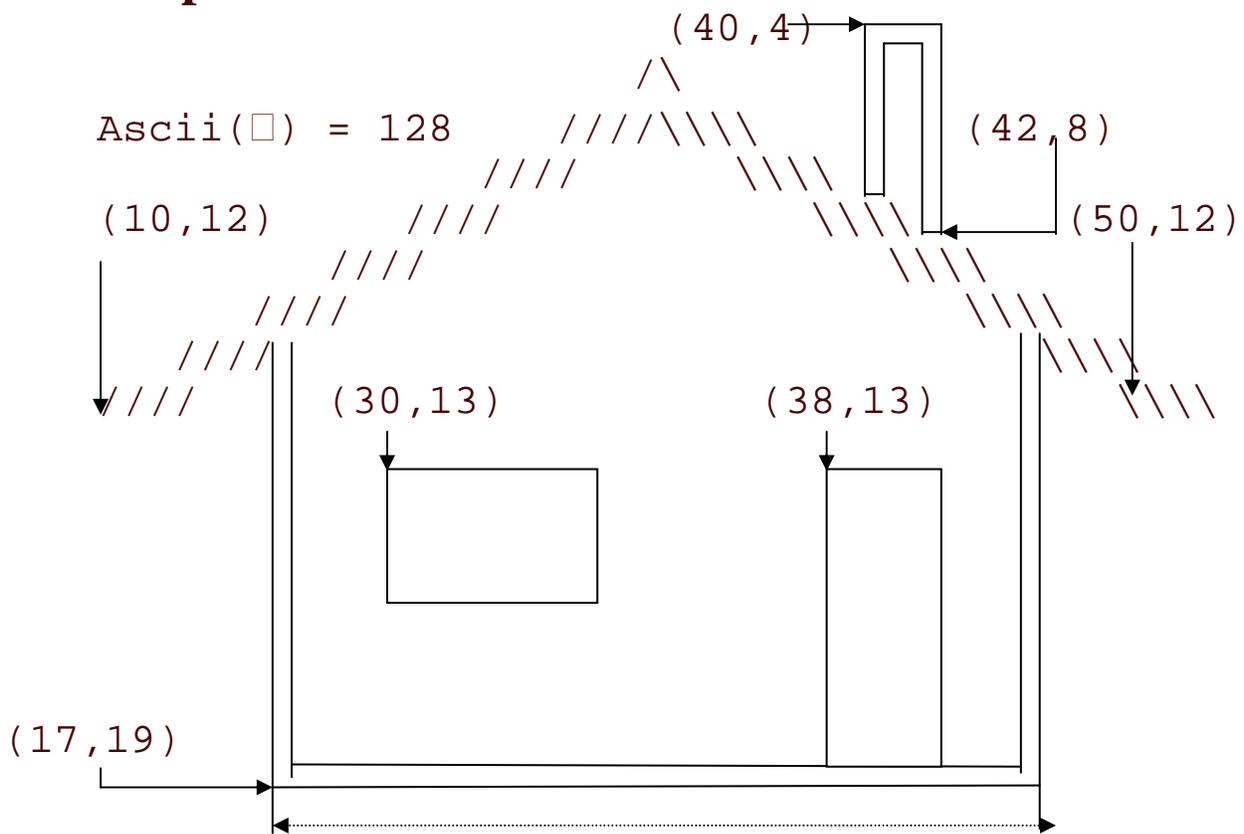
- Tabela de modos:

Modo	Significado
“r”	Abrir arquivo para leitura
“w”	Abrir arquivo para escrita
“a”	Abrir arquivo para aumentá-lo
“r+”	Abrir arquivo para leitura e escrita
“w+”	Abrir arquivo para escrita e leitura

- Neste tópico são estudados os modos “r” e “w”.
- A abertura para leitura de um arquivo que não existe falhará.
- A abertura para escrita de um arquivo não existente criará esse arquivo; se ele já existe, será apagado e reescrito.

4.5 – Controle do Cursor no Vídeo –Texto

- Tamanho máximo da tela de execução:
 - 25 linhas e 78 colunas
 - Um par (coluna, linha) exhibe 1 caractere
- Função: **gotoxy** (*expressão_1*, *expressão_2*);
 - Coloca o cursor na coluna *expressão_1* e linha *expressão_2*;
 - Está localizada no arquivo *conio.h*
- **Exemplo 4.15:** desenho de uma casa:



```

#include <stdio.h>
#include <conio.h>
main ()
{
    int i,j,x,y,z;

    /*      Desenho do telhado
                /\
            // // \\ \\
          // // \\ \\
        // // \\ \\
      // // \\ \\
    // // \\ \\
  // // \\ \\
// // \\ \\
(10,12)  (50,12)
  |         |
  v         v
  // // \\ \\
*/
for (x=10,y=12,z=40,i=1; i<=7; x+=3,y-=1,z-=6,i++) {
    gotoxy (x,y);
    printf ("%s%s", "////", z, "\\ \\ \\ \\ \\ \\");
}
gotoxy (x,y); printf ("/\\");

/*      Desenho do piso: coordenada inicial (17, 19);
        Comprimento 30 */

x=17;y=19;gotoxy(x,y);
for (i=1;i<=30;i++) printf ("%c",128);

/*      Desenho das paredes: altura 8; intervalo 28 */

for (y--,i=1;i<=8;y--,i++) {
    gotoxy(x,y);
    printf("%c%29c",128,128);
}

/*      Desenho da janela e da porta:
        janela: (22, 13)    8x3
        porta:  (38, 13)    5x6 */

for (x=22,y=13,i=1;i<=3;i++,i++){
    gotoxy (x,y);
    for (j=1;j<=8;j++) printf ("%c",128);
}

```

```

for (x=38,y=13,i=1;i<=6;y++,i++){
    gotoxy (x,y);
    for (j=1;j<=5;j++) printf ("%c",128);
}

/*          Desenho da chamine:          */

x=40;y=4;gotoxy(x,y);
for (i=1;i<=3;i++) printf ("%c",128);
for (y++,i=1;i<=3;y++,i++){
    gotoxy(x,y);
    printf("%c%2c",128,128);
}
x=42; gotoxy(x,y); printf ("%c",128);
}

```

- **Exemplo 4.16:** princípio de animação: movimentar um retângulo 3x3 em diagonal, de cima para baixo a da esquerda para a direita

```

int x, y, w, z, j, t; long i, demora=500000;
for (w=1, z=1, t=1; t<=20; w++, z++, t++) {
    gotoxy (w, z); printf (" ");
    for (x=w, y=z, i=1; i<=3; y++, i++) {
        gotoxy (x, y); printf (" ");
    }
    for (x=w+1, y=z+1, i=1; i<=3; y++, i++) {
        gotoxy (x, y);
        for (j=1; j<=3; j++) printf ("%c", 128); }
    for (i=1; i<=demora; i++);
}

```

4.6 – Vídeo Gráfico

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

void main(void)
{
    int g_driver, g_mode;
    int i, j, left, top, bottom, right;

    detectgraph(&g_driver, &g_mode);
    initgraph(&g_driver, &g_mode, "C:\\\\BC45\\\\BGI");
    left = 10; right = 600; top = 10; bottom = 200;
    for (i = top; i <= bottom; i++)
    {
        for (j = left; j <= right; j++)
            putpixel (j, i, MAGENTA);
    }
    setcolor (YELLOW);
    rectangle (left, top, right, bottom);
    getch();
    closegraph();
    return;
}
```

- O programa deve ser criado dentro de um projeto
- Deve rodar no ambiente DOS
- A biblioteca BGI deve ser incluída no “Target-Expert”