## 4. ENTRADAS E SAÍDAS

#### - scanf

Usada basicamente para promover a leitura de dados tipados via teclado.

O formato geral da função **scanf()** é:

*scanf (string-de-controle,lista-de-argumentos);*

Usando a função **scanf()** podemos ler dados do usuário. Mais uma vez, devemos ficar atentos a fim de colocar o mesmo número de argumentos que o de códigos de controle na string de controle. Outra coisa importante é lembrarmos-nos de colocar o **&** antes das variáveis da lista de argumentos, com exceção de quando for efetuada a leitura de strings.

A string de controle deve assumir os valores básicos listados a seguir:

|  |  |
| --- | --- |
| **Código** | **Significado** |
| %c | leitura de caracter  em leituras de caracteres consecutivas, para não ler <enter> da leitura anterior usar um espaço dentro das aspas 🡪 “ %c” e não “%c” |
| %d ou %i | leitura de números inteiros |
| %f | leitura de números float ou reais |
| %s | leitura de caracteres, ou seja, string  (com o scanf é possível ler 1 palavra apenas - veremos outros comandos para leitura durante essa aula)  **Nesse caso, lembre de não usar &** **antes de variável string.** |

**Exemplo:**

#include <stdio.h>

### #include <stdlib.h>

### int main()

### {

### int idade;

### float salario;

### char nome[5];

### printf("Nome:");

### scanf("%s",nome); // lê somente uma palavra

### printf("Idade:");

### scanf("%d",&idade);

### printf("Salario:");

### scanf("%f",&salario);

### 

### system("PAUSE");

### return 0;

### }

Pode –se usar o tamanho que se deseja ler, por exemplo, para 5 casas de um número inteiro usar %5d .

**- printf**

A função printf é destinada a apresentação dos dados no monitor.

Sua forma geral é

*printf (string\_de\_controle,lista\_de\_argumentos);*

Teremos, na string de controle, uma descrição de tudo que a função vai colocar na tela. A string de controle mostra não apenas os caracteres que devem ser colocados na tela, mas também quais as variáveis e suas respectivas posições. Isto é feito usando-se os códigos de controle, que usam a notação **%**. Na string de controle indicamos quais, de qual tipo e em que posição estão as variáveis a serem apresentadas. É muito importante que, para cada código de controle, tenhamos um argumento na lista de argumentos. Apresentamos agora alguns dos códigos **%**:

|  |  |
| --- | --- |
| **Código** | **Significado** |
| %d ou %i | Inteiro |
| %f | Float |
| %c | Caractere |
| %s | String |
| %% | Coloca na tela um % |

Vamos ver alguns exemplos de **printf()** e o que eles exibem:

printf ("Teste %% %%") -> "Teste % %"

printf ("%f",40.345) -> "40.345"

printf ("Um caractere %c e um inteiro %d",'D',120) -> "Um caractere D e um inteiro 120"

printf ("%s e um exemplo","Este") -> "Este e um exemplo"

printf ("%s%d%%","Juros de ",10) -> "Juros de 10%"

Para usar acentuação com o printf devemos utilizar:

#include <stdio.h>

**#include <locale.h>**

int main()

{

**setlocale(LC\_ALL, "Portuguese");**

printf("ç ã ê é");

system("PAUSE");

return(0);

}

Já vimos as constantes de barra invertida. Elas são utilizadas com o comando printf

|  |  |
| --- | --- |
| **Código** | **Significado** |
| \n | Nova linha ("new line") |
| \t | Tabulação horizontal ("tab") |
| \b | Retrocesso ("back") |
| \" | Aspas |
| \' | Apóstrofo |
| \\ | Barra invertida |
| \a | Sinal sonoro ("beep") |

**Exemplos com constantes de barra invertida**

printf(“\n\nHello!\n\n\a”); 🡪 troca de linha 2 vezes, escreve Hello, troca de linha novamente e faz bip

printf("\n\tHello!\n"); 🡪 troca de linha, pula 8 espaços, escreve Hello! e troca de linha

printf("texto sem aspas\n");

printf("\"texto com aspas\"\n");

**Exemplos de printf com variáveis**

**Exemplo1:**

int n1=10;

float n2=10.5;

printf("%d %f %.2f",n1,n2,n2);

**Exemplo2:**

char letra='A';

char nome[20]="Computacao da Unoesc";

printf("%c %s", letra, nome);

## O tamanho e precisão do campo

O tamanho do campo indica quantas posições da saída serão utilizados para exibir o valor correspondente a um especificador de formato. O tamanho do campo é determinado colocando-se um número entre o % e o caracter indicador do formato. Por exemplo, %5d indica que o valor a ser impresso vai ocupar 5 posições na tela. Por padrão, as posições em excesso são preenchidas com brancos e o valor é alinhado à direita do campo.

- Se o tamanho do campo for precedido por um **símbolo - (menos**), então o valor será alinhado à esquerda do campo.

- Se o tamanho do campo for precedido com o **símbolo 0 (zero)**, então as posições excedentes são preenchidas com zeros.

- Sempre que o tamanho do campo for insuficiente para exibir o valor solicitado, este tamanho é ignorado. Desta forma um valor nunca é truncado.

- Para valores em ponto-flutuante, pode-se definir o número de casas decimais a serem impressas, colocando-se o um ponto (.) e outro número depois do tamanho do campo. Por exemplo, o formato %10.2f, exibe o valor em **ponto-flutuante** com duas casas decimais, dentro de um campo de tamanho 10. Se esta mesma lógica for utilizada com strings (formato %s), o valor antes do ponto continua indicando o tamanho do campo, porém a segunda parte limitará o número máximo de caracteres da string a serem impressos.

Devido à forma como a função scanf trata o *buffer* (área de armazenamento temporária) de entrada, quando ela for utilizada em conjunto com as demais funções de leitura é aconselhável sempre limpar este *buffer* com fflush(stdin);

Podemos também usar a função **putchar** para imprimir um(1) caracter na tela: putchar(<variável>).

**RELEMBRANDO:**

1. Criar um algoritmo que solicita ao usuário um número inteiro, um número real e uma letra. Em seguida limpa a tela e mostra os valores lidos.
2. Criar um algoritmo que solicita ao usuário um número e depois mostra o valor da metade desse número.
3. Criar um algoritmo que calcule e mostre a área de um triângulo ((base x altura)/2). Os valores da base e da altura devem ser informados pelo usuário.
4. Faça um algoritmo que calcule e mostre a média de três números inteiros.

**EXERCÍCIOS COM scanf e printf**

* 1. Uma empresa contrata um consultor a R$1.200,00 por dia. Construir um programa que solicite o número de dias trabalhados pelo consultor e mostre o valor que deverá ser pago a ele, sabendo-se que são descontados 30% para o imposto de renda.
  2. Construir um programa que solicite ao usuário 3 números reais (float), calcule e mostre a sua média aritmética (use 2 casas decimais).

### Leitura e Escrita de Caracteres e Strings

Os caracteres são um tipo de dado: o **char**. O C trata os caracteres ('a', 'b', 'x', etc ...) como sendo variáveis de um byte (8 bits). Na linguagem C, também podemos usar um **char** para armazenar valores numéricos inteiros, além de usá-lo para armazenar caracteres de texto. Para indicar um caractere de texto usamos apóstrofes. Veja um exemplo de programa que usa caracteres:

#include <stdio.h>

int main ()

{

char Ch;

Ch='D';

printf ("%c",Ch);

return(0);

}

Muitas vezes queremos ler um caractere fornecido pelo usuário. Para isto as funções mais usadas são **getch()** e **getche()**. Ambas retornam o caractere pressionado.

**getche()** imprime o caractere na tela antes de retorná-lo e **getch()** apenas retorna o caractere pressionado sem imprimir na tela.

Também temos **getchar()** que faz a leitura de 1 caracter.

Exemplos

char Ch1, Ch2, Ch3;

Ch1=getch(); // não precisa pressionar enter

Ch2=getche(); //não precisa pressionar enter

Ch3=getchar(); //precisa pressionar enter – se forem digitados mais, pega o primeiro e ignora outros

As tabelas abaixo exibem outras funções que podem ser utilizadas para entrada/saída de dados (biblioteca conio.h e stdio.h).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Funções para char** | **Operação** | **Precisa pressionar enter** | **Mostra na tela** | **Exemplo** |
| getchar ( ) | lê um caractere | S | S | Ch1=getchar(); |
| getche ( ) | lê um caractere e mostra-o na tela | N | S | Ch2=getche(); |
| getch ( ) | lê um caractere | N | N | Ch3=getch(); |
| scanf(“%c”,&var) | lê um caractere |  |  |  |
| putchar() | Imprime um caractere na tela | - | - | putchar(Ch1); |

|  |  |  |
| --- | --- | --- |
| **Funções para strings** | **Operação** | **Exemplo** |
| gets ( ) | lê um string via teclado | gets(Str1); |
| fgets() | lê um string via teclado, controlando seu tamanho máximo | fgets(Str1, 50, stdin) |
| scanf(“%s”,var) | Lê um string até o espaço |  |
| scanf ( "%[^\n]", var) | Lê até encontrar troca de linha ou <enter> - ignora o <enter> e tudo que tiver depois |  |
| scanf ( "%256[^\n]", var); | Limita o tamanho máximo do string lido em 256 caracteres, lê até encontrar <enter> e ignora tudo que estiver depois |  |
| puts ( ) | Imprime um string na tela | puts(Str1); |

\*\*\*\*\*\*\* Lembrar de usar fflush(stdin) para limpar o buffer

Para exemplificar o uso e funcionamento das funções getchar, getche e getch:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

int main()

{

char a, b, c;

printf("Informe: ");

a=getchar();

printf("Com getchar foi lido %c\n", a);

printf("Informe: ");

b=getche();

printf("Com getche foi lido %c\n", b);

printf("Informe: ");

b=getch();

printf("Com getch foi lido %c\n", b);

system("PAUSE");

return 0;

}

IMPORTANTE: Para ler uma frase devemos utilizar gets, pois com scanf conseguimos ler somente até a primeira palavra. Porém gets permite o armazenamento além da capacidade prevista para a string. Uma solução é o uso do fgets, como no exemplo a seguir.

Exemplo:

#include <stdio.h>

#include <stdlib.h>

int main()

{

char primeiro\_nome[5];

char nome\_completo[20];

printf("Informe o primeiro nome:");

scanf("%s",primeiro\_nome);

fflush(stdin);

printf("Informe o nome completo:");

**fgets(nome\_completo,20,stdin); // ou gets(nome\_completo);**

printf("Primeiro nome = %s\n",primeiro\_nome);

printf("Nome lido = %s\n",nome\_completo);

system("PAUSE");

return 0; }