Curso Profissional: Programador/a de Informática

PSD – 10.º ano: UFCD 0809 - Programação em C/C++ - fundamentos

Ficha de Trabalho 4

Ano letivo 21/22

FORMATOS DE LEITURA e ESCRITA DE DADOS, A FUNÇÃO scanf

Formatos de leitura e de escrita de dados

Tipo	Formato
char	%c
int	%d
short int	%hd
long int	%ld
unsigned short int	%hu
unsigned long int	%lu
unsigned int	%u
float	%f ou %e ou %E
double	%f ou %e ou %E

Nota: - O tipo inteiro apresenta algumas variações, quer no tamanho, quer na forma, através dos prefixos **short, long, signed e unsigned**

Escrita de valores no ecrã usando a função printf

Depois de especificarmos os formatos de leitura na *string*, devem ser colocadas todas as variáveis correspondentes pela ordem em que ocorrem os formatos.

Exemplo - Suponhamos que, o conteúdo da variável x, de tipo real, irá ser impresso no ecrã, a sintaxe seria:

printf ("%f é um número real", x);

A função scanf

A função scanf funciona de forma semelhante à função printf, mas que permite ler dados do ecrã.

Uma vez que foi implementada para a leitura de valores, a *string* inicial deve conter apenas o formato da variável que queremos ler.

Exemplo - Suponhamos que a variável x irá receber um valor do tipo inteiro, a sintaxe seria:

```
scanf ("%d",&x);
```

Se fossem recebidos 2 valores por duas variáveis, digamos, x e y a sintaxe seria:

```
scanf ("%d %d", &x, &y);
```

Resumindo- a leitura e escrita de valores pode realizar-se através das funções **scanf** e **printf** respetivamente, utilizando os formatos adequados a cada tipo %d – int, %f – float e double, %c — char (…).
No caso do scanf, cada variável deverá ser precedida de um **&**

Programa exemplo:

```
#include <stdio.h>
main()
{
  int num;
  printf( "Introduza um n.º inteiro" );
  scanf("%d", &num);
  printf( " Número= %d e o valor do seguinte= %d\n", num, num+1);
}
```

Nota: A leitura de valores através da scanf usa o buffer do teclado como repositório temporário dos carateres que nós escrevemos, o que poderá fazer com que sejam enviados para os programas carateres indesejáveis! 😱

Um espaço em branco dentro de um scanf indica a esta função para ler e ignorar todos os espaços em branco, novas linhas e tabulações que encontrar. Por isso, quando se quer introduzir vários carateres num programa, utilizando vários scanf, para que os carateres seguintes ao 1º possam ser lidos, devemos colocar um espaço em branco, imediatamente antes do **%c** do último scanf, mas dentro da string do formato.

Outro modo de limpar todos os carateres que existem no buffer do teclado é a utilização da função: **fflush(stdin).**

Exercícios:

1. Escreve um programa que solicite, ao utilizador, dois inteiros e apresente o resultado da realização das operações aritméticas tradicionais.

2. Escreve um programa que calcule e imprima o valor de X, respeitante ao resto da divisão de dois números inteiros, Y e Z, fornecidos pelo utilizador.

3. Escreve um programa que calcule e mostre no ecrã o perímetro e a área de uma circunferência, sendo o raio fornecido pelo utilizador.

4. Escreve um programa que solicite ao utilizador a *idade*, o *montante* a depositar e o *nº de conta* em que se realiza o depósito. Declara as variáveis como *short int*, float e *long* int. Deve ser impresso no ecrã os dados inseridos, em linhas diferentes.

5. Indica quais as declarações estão incorretas. Justifica.

   A. y int;
   B. int ;
   C. integer x;
   D. inta, b;
   E. float f, g, h;
   F. char ch1=ch2='A';
   G. char ch1='A', ch2='A';

7. Indica, na seguinte lista, os identificadores corretos e incorretos de variáveis.
   A. Valor
   B. &dez
   C. _underscore
   D. 10b
   E. main
   F. f1

8. Indica quais das seguintes afirmações são verdadeiras e quais são falsas, CORRIGINDO AS FALSAS.
   A. O tipo **float** reserva espaço em memória para um real com precisão simples e o tipo **double** reserva espaço para um real com precisão dupla.
   B. Uma variável do tipo **char** pode armazenar carateres individuais ou strings.
   C. Uma variável pode armazenar vários carateres desde que sejam carateres especiais.
   D. O operador **%** não pode ser utilizado em reais.

9. Escreve um programa que solicite ao utilizador uma determinada data (dia, mês e ano) e mostre em seguida no ecrã, esses dados no formato dd/mm/aaaa.

Operador **sizeof** – operador que indica o nº de bytes ocupados por um determinado tipo de expressão.

Sintaxe: sizeof <expressão> ou sizeof(<tipo>)

Exemplo:

```
printf (“ O tamanho em bytes de um inteiro = %d\n”, sizeof(int));
```

Também o iremos usar mais tarde na determinação dos bytes ocupados pelos elementos de uma estrutura (struct), poder pesquisar um determinado registo num ficheiro ou o tamanho de um ficheiro de carateres.