Ficha de trabalho
Introdução à linguagem C

V1.1 - V.Lobo, Escola Naval, 2001

NOTA PRÉVIA: Para fazer os primeiros programas em C terá que escrever no ecrã, e ler do teclado. Como essa matéria será dada mais tarde, apresenta-se aqui uma maneira simples de o fazer:

1. Para escrever no ecrã, pode-se usar a função ***printf***. A função ***printf*** recebe um número variável de parâmetros: o primeiro é sempre a ***string de formatação***, que especifica como é que se vai escrever; os restantes parâmetros são as variáveis a escrever. Na ***string de formatação***, podem estar caracteres (que serão impressos tal como estão no código), e ***especificadores de formato***, que indicam que se deve imprimir o valor de uma variável. Os dois ***especificadores*** que vamos necessitar são o %d para imprimir valores inteiros, e %f para valores reais (float). Exemplos:

Código	O que aparece no ecrã	
printf("ola.\n")	ola	
printf("%d \n",xpto)	3	(se xpto tiver o valor 3)
printf("o valor de b é %d \n",y)	o valor de b é 37	(se b tiver o valor 37)

2. Para ler do teclado, pode-se usar a função scanf. A função scanf tem dois parâmetros: o primeiro é a ***string de formatação***, que indica como será lido o valor (%d para ler em formato inteiro, %f para ler em formato de vírgula flutuante), e o segundo será a ***variável*** onde o valor será guardado, ***precedido pelo símbolo &***. Exemplos:

Código	O que faz
scanf("%d", &xpto)	Lê um inteiro para a variável xpto
scanf("%f", &y)	Lê um valor real para a variável y

1) Faça um programa que calcule o volume de um paralelepípedo de lados 37, 20, e 17.

2) Faça um programa para converter escudos em euros, ou vice-versa (o programa deverá permitir fazer qualquer das duas opções).

3) Faça um programa que calcule as raízes de equações do 2º grau. Teste o programa com os valores a=1,1,10; b=0,1,2; c=0,-10,20.
   2.1) Faça um programa que ignore raízes imaginarias
   2.2) Faça um programa que dê também as raízes imaginárias.

4) Escreva uma rotina para converter coordenadas cartesianas em coordenadas polares, e vice-versa..

5) Num grande prémio de fórmula 1, o vencedor recebe 10 pontos, o segundo classificado 6, o terceiro 4, e os três seguintes 3,2, e 1 ponto. Os restantes concorrentes não recebem pontos.

5.1) Faça um programa que pergunte ao utilizador a posição de um concorrente, e lhe diga quantos pontos tem.

5.2) Faça um programa que pergunte em que lugar é que um dado concorrente ficou em cada uma das provas do campeonato, e no fim lhe diga o número total de pontos, o número de provas em que pontuou, e os pontos médios por prova.

6) A série de Fibonacci é uma série em que cada elemento é a soma dos dois anteriores. A série é inicializada com dois 1, sendo por isso 1,1,2,3,5,8,13,21….. Escreva um programa que imprima a série de Fibonacci até um dado número limite, introduzido pelo utilizador.

7) Uma das primeiras áreas de aplicação das máquinas de cálculo foi a artilharia. Em memória disso escreva programas para calcular a elevação que o cano deve ter para atingir um alvo, dada uma velocidade inicial e uma distância a esse alvo, partindo do princípio que não há atmosfera e que a bala é um ponto material (ou seja usando condições ideais).

8) Escreva uma rotina para ordenar vectores de números inteiros. A rotina deverá receber como parâmetro um vector de inteiros, e a sua dimensão (guardada noutro inteiro). A rotina deverá devolver o vector ordenado.

8.1) Escreva duas rotinas diferentes de ordenação (por exemplo, um bubble sort simples e um quicksort). Escreva um programa para testar a eficiência das duas rotinas. Tenha em conta que convém testar as rotinas com diferentes tipos

de vectores (vectores com valores aleatórios, vectores quase-ordenados, etc), e convém fazer um número de testes que seja estatisticamente significativo.

9) Em vários campos da ciência e tecnologia (por exemplo em criptografia) é importante saber se um dado número é ou não primo.

9.1) Escreva uma rotina que dado um número inteiro, verifique se esse número é ou não primo.

9.2) Escreva um programa que escreva no écrã todos os números primos menores que um numero dado pelo utilizador.

10) Uma das técnicas de criptografia mais antigas é atribuída aos Romanos, e o seu nome, cifra de césar, vem de um dos seus grandes generais; Júlio César. A cifra de césar consiste simplesmente em substituir cada letra de uma mensagem pela letra que está *n* posições à sua frente (ou atrás) no alfabeto. Ao deslocamento *n*, que é usado quer para passar a mensagem para cifra quer para a reconverter em texto claro, chama-se "chave". A palavra "adeus", cifrada com a chave 2 seria convertida em "cfgwu". Se o deslocamento passar para além da última letra, regressa ao início do alfabeto ('z'+1='a'). Embora haja bastantes variantes, vamos supor que na cifra de césar que vamos usar, apenas as letras do alfabeto (incluindo k,w,y) são cifradas, ficando os restantes símbolos (incuindo os espaços em branco e mudanças de linha) tal como se encontram no texto original. Para além disso, qualquer letra maiúscula é convertida em minúscula antes de ser cifrada.

10.1) Escreva uma rotina que, dada uma chave e um texto (numa string), cifre o texto de acordo com as regras dadas.

10.2) Os ficheiros "ordem.txt" e "sistemas.txt" contêm, respectivamente, uma ordem do dia à Escola Naval, e informações sobre a próxima repetição escrita de Sistemas Digitais II. Infelizmente, esses textos estão encriptados com cifra de césar, e não conhecemos as chaves. Escreva um programa que, "por força bruta", descubra qual a chave usada, e descodifique os textos. Para detectar automaticamente a chave correcta, lembre-se que um texto sobre Sistemas Digitais II conterá provavelmente a palavra "linguagem" ou "microprocessador", e conhece bem o texto típico de uma ordem do dia à Escola Naval.

11) Escreva um conjunto de rotinas que implemente um stack para guardar fichas que têm um inteiro chamado "cota", e uma string com um máximo de 64 caracteres chamada "nome". Deverá ter uma rotina chamada Init_stack() para inicializar o stack, outra chamada Push_stack() para fazer o "push", outra chama Pop_stack() para fazer o "pop", e finalmente uma chamada Clear_stack(), para eliminar o stack, libertando a memória que este ocupava. Escreva um parágrafo apenas como "manual de instruções" para essas rotinas.