Exercícios introdutórios

(a)"Hello World!": Escreva em C programa que quando executado coloca a string "Hello World!" no écran. Acrescente um comentário ao programa documentando-o e recompile-o.

(b)"Soma de 2 números inteiros": Escreva um novo programa em C que soma dois números inteiros, previamente declarados e inicializados respectivamente com 7 e 9, e escreve o resultado da soma.

(c)"Relatório em LaTeX: Com a ajuda do docente, escreva um pequeno relatório em LaTeX documentando os dois exercícios que fez.

Problemas de cálculo numérico

(a) Escreva um programa em C que leia do teclado uma informação horária válida, composta por horas, minutos e segundos, e escreva no monitor o tempo total em segundos.

(b) Escreva um programa em C que, dada uma temperatura em graus Celsius, que é lida do teclado, a converta para graus Fahrenheit e a escreva no monitor. A fórmula de conversão é: **F=1.8*C+32**.

(c) Escreva um programa em C que, dado um tempo em segundos lido do teclado, escreva no monitor o tempo com o formato **hh:mm:ss**. Considere que, para fazer a divisão inteira, existe o operador '/' e, para fazer o resto da divisão inteira, existe o operador **%**.

Problemas com estruturas de controlo

(a) Desenvolva o algoritmo, e posteriormente codifique-o em C, de um programa que lê dois inteiros e escreve o maior deles.

(b) Desenvolva o algoritmo, e posteriormente codifique-o em C, de um programa que lê N inteiros e escreve o maior deles. O programa deverá terminar a leitura de números inteiros quando for introduzido o número 0.

(c) Escreva um programa em C, que lê um carácter e imprime "vogal" ou "consoante" de acordo com o carácter introduzido (note que programa deverá funcionar com maiúsculas e minúsculas).

(d) Escreva um programa em C, que lê um número e imprime "PAR" ou "IMPAR" conforme o caso.

(e) Escreva um programa em C, que calcula o módulo de um número.

(f) Escreva um programa em C, que lê um determinado número inteiro N e escreve no monitor os números pares até N.

(g) Escreva um programa em C, que lê 2 números inteiros, **a** e **b**, e escreve o resultado de elevar **a** à potência de **b** (utilize estruturas cíclicas para calcular o valor da potência).

(h) Escreva um programa em C, que lê um inteiro positivo e escreve o valor do seu factorial (crie duas versões: uma recursiva semelhante à que conhece da Programação Funcional e outra utilizando um ciclo **while**).

(i) Escreva um programa em C que leia do teclado um número real, um carácter que representa uma operação aritmética **(+, -, /, *)** e outro número real, e que faça a operação aritmética pretendida e coloque o resultado no monitor. Tenha em atenção que o computador não faz divisões por 0. Trate esta excepção. O resultado deverá ter a seguinte forma: **operando1 operação operando2 = resultado**.

(j) Considere a seguinte fórmula que relaciona graus Celsius com graus Fahrenheit fornecida no primeiro exercício. Escreva um programa em C, que calcula uma tabela de conversão entre graus Celsius e graus Fahreneit com Fahrenheit a variar entre 0 e 300 (considere intervalos de 5 graus).

Pequenos problemas

Para cada um dos problemas seguintes tente especificar uma pequena função em C isolando o algoritmo, a seguir escreva um programa principal onde essa função é invocada para produzir resultados.

Tente escrever duas versões da função para cada problema: uma iterativa e outra recursiva.

(a) Especifique uma função em C, que recebe um carácter e dá como resultado 0 ou 1 conforme o carácter seja consoante ou vogal.

(b) Especique uma função, que calcula o módulo de um número.

(c) Especifique uma função, que recebe 2 argumentos inteiros, **a** e **b**, e dá como resultado **a** elevado à potência de **b**.

(d) Especifique uma função, que recebe um inteiro positivo e devolve o valor do seu factorial.

(e) Especifique uma função que recebe as coordenadas de dois pontos cartesianos e devolve a distância real entre eles (terá de usar a biblioteca "math.h" e compilar o programa com "-lm").

(f) Especifique uma função que terá dois argumentos inteiros e produzirá um resultado inteiro que é o menor múltiplo comum dos argumentos recebidos. Posteriormente codifique um programa principal (main()) que utiliza esta função para cálculo do menor múltiplo comum de dois números fornecidos pelo utilizador.

(g) Especifique um programa que lê dois valores inteiros, correspondentes ao numerador e ao denominador de uma fracção, e que produz como resultado a fracção reduzida correspondente.

(h) Especifique um programa que lê quatro valores inteiros, correspondentes aos numeradores e denominadores de duas fracções, e que produz como resultado a fracção resultante da soma das fracções introduzidas.

Sequências de números inteiros positivos

Especifique um programa e codifique-o em C que lê do teclado uma sequência de números inteiros positivos e calcula um conjunto de indicadores sobre a sequência lida. O programa deverá oferecer ao utilizador um menu de operações como o que se mostra a seguir:

```
    Sequências de Inteiros: lista de operações
A - Ler a sequência
B - Escrever a sequência
C - Calcular o máximo da sequência
D - Calcular o mínimo da sequência
E - Determinar a subsequência de números acima da média
F - Determinar a subsequência de números abaixo da média
G - Calcular o mínimo múltiplo comum da sequência
H - Determinar a subsequência dos números que são primos
I - Sair do Programa
     Opção:
```

A leitura de números deverá terminar quando fôr lido o 0 ou quando fôr atingido o limite da sequência (defina-o).

Apresente o resultado de forma a facilitar a análise.

Sugestão de implementação: especifique uma função para cada opção do menu inicial.

Totoloto (retirado do livro de apoio)

Pretende-se escrever um programa que leia do teclado uma chave de totoloto e que a imprima no monitor segundo as regras convencionais. Admita, para simplificar, que a combinação de números, que forma a chave, tem obrigatoriamente que ser introduzida por ordem crescente. A entrada de dados deverá ter a seguinte forma:

```
Elemento 1 da chave = ...
Elemento 2 da chave = ...
...
Elemento 6 da chave = ...
```

Admitindo que a chave introduzida foi {2, 12, 17, 27, 30, 43}, a saída de resultados deve obedecer ao formato seguinte:

```
          Aposta de Totoloto
 1     X     3     4     5     6     7
 8     9    10    11     X    13    14
15    16     X    18    19    20    21
22    23    24    25    26     X    28
29     X    31    32    33    34    35
36    37    38    39    40    41    42
 X    44    45    46    47    48    49
```

Sugestão: relembre o formato de escrita formatada de strings discutido nas aulas teóricas (%ns).

Análise de Strings (adaptado do livro de apoio)

Pretende-se escrever um programa que leia do teclado uma linha de texto até ao máximo de 60 caracteres, e que calcule: o número de vogais, o número de consoantes, o número de caracteres minúsculos, o número de caracteres maiúsculos e o número de caracteres numéricos.

A entrada de dados deve ter a seguinte forma:

```
Frase de entrada: ######################
```

A saída de dados deve ter a seguinte forma:

```
Frase de Entrada: ######################
Número de vogais: ##
Número de consoantes: ##
Número de caracteres minúsculos: ##
Número de caracteres maiúsculos: ##
Número de caracteres numéricos: ##
```

Implemente funções para a leitura, escrita e para cada um dos indicadores estatísticos.

Sugestão: relembre/investigue as funções da biblioteca **ctype.h**. Estas funções têm a seguinte assinatura:

```
int nome_função( int carácter )
```

O valor devolvido é nulo (em C é falso) se o carácter passado como argumento não pertence à classe) e não nulo (em C é verdadeiro) no caso contrário.

Para este exercício, as funções relevantes são:
isalpha
caracteres alfabéticos

isdigit
digitos decimais

islower
letras minúsculas

isupper
letras maiúsculas

Ficha Nº3 Exercício Nº4 2008.03.10

Capicua

Especifique uma função que recebe uma string e verifica se a mesma é capicua (lida da esquerda para a direita, ou da direita para a esquerda, é a mesma palavra/frase).

Exemplos típicos: ala ou rapar.

Um caso curioso (e, claro, mais complexo) é a seguinte Capicua brasileira: Socorram-me subi no onibus em Marrocos.

Ficha Nº3 Exercício Nº5 2008.03.10

Conversão Romano-Árabe

Especifique uma função que recebe uma string contendo um número escrito em notação romana e que dá como resultado o respectivo valor inteiro (correspondente à notação arábica).

Cifra por substituição arbitrária

Considere a cifra (palavra-chave) constituída pelas letras "SAPO" seguidas de um dígito n (entre 0 e 9). Para a cifragem (codificação) da mensagem efectuam-se os seguintes passos:

1. reduzem-se espaços seguidos a um único espaço e mantém-se os caracteres de pontuação;
2. os digitos rodam-se (circularmente) para a direita da quantidade n;
3. convertem-se todas as letras em maiúsculas ou minúsculas;
4. considera-se a seguinte tabela de codificação: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z S A P O B C D E F G H I J K L M N Q R T U V W X Y Z

A construção desta tabela limita-se a colocar a palavra chave no início e completar com as restantes letras do alfabeto, não colocando as letras que se repetem.

Para a codificação, substitui-se cada um dos carácteres da primeira linha pelo carácter correspondente da segunda linha.

Construa um programa que dada uma cifra, codifique a mensagem, e um outro que, dada a mensagem cifrada e a cifra, descodifique a mensagem.

Ficha Nº4 Exercício Nº1 2008.03.31

Gestão de Armazens

Uma empresa tem 10 armazens e precisa de um programa que leia as vendas mensais dos 10 armazens, calcule a media de vendas e produza uma listagem dos armazens com vendas superiores à média calculada.

Ficha Nº4 Exercício Nº2 2008.03.31

Contem ou Contido

Desenvolva um algoritmo e posteriormente codifique-o em C que dadas dumas matrizes de dimensões arbitrárias verifica se a primeira contem a segunda (considere que os valores armazenados nas matrizes são do tipo inteiro).

Média dos Alunos

Cada aluno de uma licenciatura em Ciências da Computação pode ter notas correspondentes a 10 disciplinas feitas durante o ano lectivo (caso não tenha sido avaliado positivamente a uma disciplina não terá essa nota).

(a) Declare as estruturas de dados (o modelo) necessárias para suportar o sistema de informação: um aluno tem um número (inteiro sequencial a partir de 1), um nome (string) e uma lista de notas (reais). Pense na hipótese de encarar o número do aluno como um código alfanumérico (para permitir números em qualquer escala, ou mesmo o uso de letras) e identifique as alterações daí decorrentes.

(b) Inicialize a estrutura de dados: na declaração e/ou numa função de inicialização que é chamada no início da função **main()**.

(c) Crie as funções normais de manipulação de uma pequena base de dados:

1. Inserir - de modo a manter a informação por ordem alfabética de nome. Numa 1ª versão, utilize o algoritmo por trocas directas e, numa 2ª versão, use a inserção ordenada;
2. Remover - utilize a pesquisa sequencial ordenada para encontrar o registo a remover;
3. Consultar - utilize a pesquisa dicotómica para encontrar o registo a consultar;
4. Listar - liste todos os alunos armazenados, por ordem alfabética.

(d) Especifique uma função em C para calcular a média de cada aluno (faça uma função que dada a estrutura de dados principal e o identificador do aluno, produz como resultado a sua média); note que disciplinas não avaliadas não contam para a média (diferente de obter a classificação 0 numa disciplina).

(e) Calcule a média geral da turma.

(f) Calcule a média da turma em cada disciplina.

(g) Calcule a percentagem de faltas (ausência de nota).

Rectângulos

No plano cartesiano um rectângulo com os lados paralelos aos eixos pode ser univocamente determinado por uma diagonal dada por dois pontos. Assim, para representar esta figura geométrica, definiu-se em C o seguinte tipo de dados:

```
typedef struct sPonto
  {
          float x;
          float y;
  } Ponto;

typedef struct sRectangulo
  {
          Ponto p1;
          Ponto p2;
       } Rectangulo;
```

Especifique as seguintes funções e utilize-as num programa.

```
float Area( Rectangulo r )
{
  ...
}

float Perimetro( Rectangulo r )
{
  ...
}
```

Jornadas de Futebol

Pretende-se guardar a informação sobre os resultados dos jogos de uma jornada de um campeonato de futebol na seguinte estrutura de dados:

```
typedef int Golos;

typedef char Equipa[30];

typedef struct sInterv
  { Equipa e;
    Golos g;
   }  Interv;

typedef struct sJogo
  { Interv i1, i2; }  Jogo;

typedef Jogo Jornada[20];
```

Desta forma podemos ter um programa que vai trabalhar as jornadas e que declara as seguintes variáveis:

```
...
int main()
{
  Jornada j1, j2;
  Jornada campeonato[56];
  ...
}
```

Depois de analisar bem a estrutura de dados especifique as seguintes funções e crie um programa para as testar.

```
bool igualj( Jornada j );
/* que verifica se nenhuma equipa joga com ela própria */
bool semrepet( Jornada j );
/* que verifica se nenhuma equipa joga mais do que um jogo */
??? empates( Jornada j );
/* que dá a listas dos pares de equipas que empataram na jornada */
??? equipas( Jornada j );
/* que dá a lista das equipas que participam na jornada */
??? calcres( Jornada j );
/* que calcula os pontos que cada equipa obteve na jornada (venceu - 3
pontos; perdeu - 0 pontos; empatou - 1 ponto) */
```

Como exercício extra defina tipos de dados para suportar o tipo de resultado de algumas funções (aquelas que têm o tipo a ???).

Polinómios

Uma forma de representar polinómios de uma variável é usar listas de pares (coeficiente, expoente).

Em Haskell a definição do tipo de dados seria:

```
type Pol = [(Float,Int)]
```

Note que o polinómio pode não estar simplificado. Por exemplo,

```
[(3.4,3), (2.0,4), (1.5,3), (7.1,5)] :: Pol
```

representaria o polinómio: 3.4 x3 + 2 x4 + 1.5 x3 + 7.1 x5.

Desenvolva então, em C, as seguintes alíneas:

(a) Defina em C as estruturas de dados para suportar os polinómios.

(b) Defina a função de cálculo do valor de um polinómio num ponto.

(c) Defina uma função que dado um polinómio, calcule o seu grau.

(d) Defina uma função que calcule a derivada de um polinómio.

(e) Defina uma função para ordenar um polinómio por ordem crescente de grau.

(f) Defina uma função para simplificar um polinómio.

(g) Defina uma função para somar dois polinómios nesta representação.

(h) Defina uma função que calcule o produto de dois polinómios.

Imagens PNM

Considere que uma imagem é representada por uma matriz. Cada *pixel* é uma célula dessa matriz. Para que seja possível representar cor optou-se por definir cada célula como um array de três posições que guardam as componentes vermelha, verde e azul da côr.

Defina funções que, dadas duas coordenadas, e uma côr:

(a) Desenhe um rectângulo,

(b) Desenhe um rectângulo preenchido,

(c) Desenhe uma recta que una os dois pontos,

(d) Defina uma função que exporte a imagem num formato PNM (consulte a Wikipedia para uma descrição sucinta dos formatos disponíveis),

(e) Defina uma função que dado um ponto, uma côr, e uma medida (raio), desenhe uma circunferência.

Listas em Memória "Estática"

Relembre o exercício realizado na última aula teórica onde se implementou uma lista de alunos através de um **array** em C. Os alunos são armazenados pela ordem em que chegam no entanto o acesso à informação quer-se feito por ordem alfabética de nome. Para isso, acrescentou-se um campo adicional ao aluno que permite criar uma ordenação lógica entre alunos.

Os tipos de dados definidos na aula para modelar o problema foram os seguintes:

```
#define MAXALUNO 100

typedef struct sAluno
  {
    char * nome;
    int idade;
    int prox;
  } Aluno;

typedef struct sListAluno
  {
    int cabeca; /* -- cabeça da lista -- */
    int livre;  /* -- primeira posição livre -- */
    Aluno lista[MAXALUNO];
    } ListAluno;
```

Tomaram-se as seguntes decisões:

- A lista está vazia quando **cabeca = livre**;
- O último elemento da lista tem o campo próximo com o valor **-1**.
- O campo **nome** está declarado como apontador, logo terá de ser alocado espaço para nele guardar informação; como se trata de uma string use a função **strdup**.

Depois de analisar bem os tipos de dados definidos (faça um esquema com alguns elementos de informação), atente no seguinte esqueleto de programa:

```
int main()
  {
    Aluno a1 = {"Carlos", 17, -1},
          a2 = {"Ana", 19, -1},
          a3 = {"Zulmira", 18, -1},
          a4 = {"Paulo", 18, -1},
          a5 = {"David", 19, -1};
    ListAluno l1;
    int i;

    l1 = initLista(l1);
    l1 = insereAluno(l1, a1);
    l1 = insereAluno(l1, a2);
    l1 = insereAluno(l1, a3);
    l1 = insereAluno(l1, a4);
    l1 = insereAluno(l1, a5);

    listAluno(l1);
    i = procuraAluno(l1, "Ana");
    consultAluno(l1, i);
    }
```

Desenvolva, ao longo das alíneas seguintes, as funções necessárias para colocar o programa a funcionar:

(a)A função de inicialização que deverá inicializar uma estrutura do tipo **ListAluno**:

```
ListAluno initLista(ListAluno l);
```

(b)A função de procura que dada uma lista e um nome dá como resultado o índice onde esse aluno se encontra na lista (devolve -1 se o aluno não pertencer à lista):

```
int procuraAluno(ListAluno l, char* nome);
```

(c)A função de consulta que dada uma lista e um índice, escreve no monitor a informação relativa ao aluno armazenado nesse índice:

```
void consultAluno(ListAluno l, int indice);
```

(d)A função de inserção que recebe uma lista e um aluno e coloca o aluno na lista actualizando o índice (campo próximo) e devolvendo uma nova lista alterada:

```
ListAluno insereAluno(ListAluno l, Aluno a);
```

(e)A função de listagem que recebendo uma lista produz uma listagem ordenada alfabeticamente por nome dos alunos presentes na lista:

```
void listAluno(ListAluno l);
```

(f)A função de remoção que dada uma lista e um nome remove o respectivo aluno da lista devolvendo a lista alterada:

```
ListAluno removeAluno(ListAluno l, char *nome);
```

Ficha Nº8 Exercício Nº1 2008.04.14

Lista de Inteiros

Considere uma lista de inteiros (não se sabe o seu comprimento). Especifique então as seguintes funções e estruturas de dados:

(a) Defina os tipos necessários para suportar uma lista ligada de inteiros.

(b) Especifique uma função para inserir um valor na cabeça da lista.

(c) Especifique uma função para listar os valores da lista, do início para o fim (faça também a função que lista os elementos na ordem inversa).

(d) Especifique uma função para procurar um valor na lista (como resultado deverá devolver um apontador para o elemento ou NULL caso não o encontre).

(e) Especifique uma função para contar os elementos da lista.

(f) Especifique uma função para calcular o maior elemento na lista.

(g) Especifique um programa, usando as funções definidas, que cria uma lista com os múltiplos de 3 entre 0 e 100 e os lista por ordem decrescente e crescente.

Ficha Nº8 Exercício Nº2 2008.04.14

À procura da saída ...

Suponha que existe um labirinto (pense numa representação adequada para o mesmo) que tem um ponto de entrada e um ponto de saída. Especifique um algoritmo que vai descobrir um caminho possível entre o ponto de entrada e o ponto de saída. Ajuda: modele numa lista ligada uma stack para ir guardando o caminho percorrido e as hipóteses alternativas.

A Agenda de Contactos

Pretende-se que desenvolva uma aplicação para gerir uma agenda de contactos. Uma agenda é uma lista de entradas ou grupos de entradas. Uma entrada tem a seguinte constituição:

chave
: chave única de identificação (não pode haver duas entradas com a mesma chave);

tipo
: tipo da entrada: pessoa, empresa, ...

nome
: nome da pessoa, empresa ou entidade;

email
: contacto electrónco (é opcional)

telefone
: número de telefone (obrigatório)

Um grupo pode ter entradas, referências a entradas já existentes na agenda (por chave) ou subgrupos (os grupos podem ter grupos aninhados infinitamente). O grupo tem, então, a seguinte constituição:

chave
: chave única de identificação (não pode haver dois grupos com a mesma chave);

nome
: nome do grupo; lista de itens: entradas e/ou grupos e/ou referências;

Por sua vez, a referência é apenas constituída pela chave da entrada ou grupo que referencia.

Desenvolva a aplicação nas seguintes etapas:

(a) Defina as estruturas de dados necessárias para suportar o sistema de informação;

(b) Especifique as várias funções de inserção: inserir uma entrada na agenda, inserir um grupo na agenda, inserir uma entrada num grupo, ...

(c) Especifique uma função para listar o conteúdo da agenda.

(d) Especifique uma função para gravar o conteúdo da agenda num ficheiro.

(e) Especifique uma função para carregar o conteúdo da agenda de um ficheiro.

(f) Especifique as várias funções de remoção.