

Aula prática #11 – Registos

Problema 1

Escreva e teste uma biblioteca de funções chamada `polinomios` para operar com polinómios. Considere que os coeficientes dos polinómios são inteiros e que o grau máximo é 10.

a) Implemente uma função de leitura de polinómios com a seguinte assinatura:

```
1 int lerPoli(int *poli)
```

b) Implemente uma função de escrita de polinómios com a seguinte assinatura:

```
1 void escrevePoli(int *poli, int grau)
```

c) Implemente uma função que adicione 2 polinómios com a seguinte assinatura:

```
1 void adicionaPoli(int *p1, int *p2, int *pRes, int maiorGrau)
```

d) Implemente uma função que calcule o valor do polinómios para um certo valor da variável independente com a seguinte assinatura:

```
1 float calc(int *poli, int grau, float x)
```

e) Implemente um programa que permita testar a biblioteca desenvolvida.

Problema 2

Escreva e teste um programa que para uma data completa de um certo dia (guardada num registo), indique a data de k dias à frente ou atrás do dia especificado. Utilize a biblioteca "datas" e a estrutura sugerida para o programa disponível no arquivo prob11-2.zip.

Conteúdo do ficheiro prob11-2.zip:

- `datas.h`: Cabeçalho (header file) com protótipo de todas as funções necessárias para este exercício.
- `datas.c`: Implementação de grande parte das funções da biblioteca. Deverá implementar as funções `diaSeguinte`, `diaAnterior` e `somaDias`.
- `programa.c`: Ficheiro onde reside o programa principal (função `main()`), que deve ser completado para resolução do exercício.
- `Makefile`: ficheiro que permite compilar automaticamente todos os ficheiros no programa final.

Como compilar:

- Opção 1: `clang datas.c programa.c -o programa`
- Opção 2 (usando o ficheiro `Makefile`): `make`

Exemplo

```
1 data (dia mes ano)? 3 6 2016
2 numero de dias? 2
3 5 6 2016
4 data (dia mes ano)? 15 12 2016
5 numero de dias? -2
6 13 12 2016
7 data (dia mes ano)? 25 12 2016
8 numero de dias? 10
9 4 1 2017
```

Problema 3

Os números complexos são representados por pontos no plano, designando-se uma componente (normalmente segundo a coordenada horizontal) por parte real, e a outra componente por parte imaginária do complexo.

Escreva uma biblioteca de rotinas chamada `complexos` para operar com complexos.

Um complexo z com componentes x (parte real) e y (parte imaginária) escreve-se $z = x + iy$; dados os complexos $u = a + ib$ e $v = c + id$ a soma é $u + v = (a + c) + i(b + d)$; o módulo e argumento angular de z correspondem ao módulo e argumento angular do vector da origem a (x, y) .

a) Defina um registo que permita guardar um numero complexo.

b) Implemente uma função de leitura de números complexos:

```
1 complexo leComplexo()
```

c) Implemente uma função de escrita de números complexos:

```
1 void escreveComplexo(complexo c)
```

d) Implemente uma função que adicione 2 números complexos:

```
1 complexo somaComplexo(complexo c1, complexo c2)
```

e) Implemente uma função que calcule o modulo de um numero complexo:

```
1 double modComplexo(complexo c)
```

f) Implemente uma função que calcule o argumento angular de um numero complexo:

```
1 double argComplexo(complexo c)
```

g) Escreva um programa que permita testar a biblioteca que desenvolveu.

Problema 4

Uma empresa de electrodomésticos pretende informatizar os dados relativos aos artigos de que dispõe para venda.

- a) Defina um registo, artigo, adequado à representação de um artigo, contendo a seguinte informação: designação do artigo (televisor, rádio, máquina de lavar roupa, etc.), marca (por exemplo, Blabla), modelo (por exemplo, S-30), preço e quantidade disponível em armazém.
- b) Escreva uma função que leia um elemento do tipo artigo, passado por parâmetro. Os dados do artigo são lidos a partir da informação introduzida pelo utilizador.

```
1 void leArtigo(artigo *item)
```

- c) Considere que a informação sobre os artigos da empresa, em número $n \leq 10000$, foi lida para um vetor (armazém) de elementos do tipo artigo. Escreva uma função que retorne o número total de artigos de uma certa marca e modelo a especificar.

```
1 int totalArtigos(artigo armazem[], int n, char marca[], char modelo[])
```

- d) Implemente um procedimento que liste todos os produtos cuja existência em stock é inferior a 10 unidades

```
1 void alertaArtigos(artigo armazem[], int n)
```

- e) Implemente um procedimento que ordene todos os produtos em stock pela sua quantidade.

```
1 void ordenaArtigos(artigo armazem[], int n)
```

- f) Implemente um programa que permita testar a biblioteca que desenvolveu.

Problema 5

Pretende-se implementar um jogo de cartas apresentado no ficheiro `jogo.c`. O jogo é o seguinte: os jogadores vão retirando cartas do baralho e comparam o valor da carta. Aquele que vencer essa ronda ganha 1 ponto e quem perdeu a ronda, perde um ponto. Em caso de empate, as pontuações não se alteram. Os jogadores iniciam com 20 pontos cada um. O jogo termina quando se atingem as 100 rondas ou quando um dos jogadores fica com zero pontos.

A lógica do jogo está já parcialmente implementada no ficheiro `jogo.c`. Complete as seguintes funções que ainda não foram implementadas:

```
1 void criaBaralho(cartas *baralho)
```

Deve criar um conjunto de cartas que respeite as estruturas de dados definidas. O baralho deve ter uma carta de tipo para todos os naipes.

```
1 void shuffle(cartas *baralho)
```

Baralha as posições das cartas dentro do baralho de forma aleatória.

```
1 int comparaCarta(cartas c1, cartas c2)
```

Verifica se uma carta é maior do que outra. O retorno da função deve ser 1, 0 ou -1 conforme a carta `c1` seja maior, igual ou menor do que a carta `c2`, respectivamente.