



Programação 2

2º Semestre

Trabalho Prático P2B

Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Grupo 6

Pretende-se fazer um programa para adivinhar o tipo de um ponto qualquer do plano. O tipo de um ponto pode ser 'GOOD', 'FAIR' e 'POOR' (pretende traduzir alguma medida de qualidade).

Considere as classes CPoint e Guesser:

```
class CPoint
{
    double x;
    double y;
public:
    CPoint(double a, double b):x(a), y(b) {}
};

class Guesser //adivinho
{
public:
    enum POINTTYPE {GOOD=0, FAIR, POOR};

    virtual POINTTYPE guessTypeOfSinglePoint (CPoint& pt) = 0;
    virtual vector<POINTTYPE> guessTypeForSetOfPoints (vector<CPoint>& vpt);
};
```

a) Implemente o membro-função de Guesser

```
vector<POINTTYPE> guessTypeForSetOfPoints (vector<CPoint>& vpt);
```

que adivinha aleatoriamente (com probabilidade uniforme) o tipo para o vector de pontos pedido. Esta função deve usar a função

```
POINTTYPE guessTypeOfSinglePoint (CPoint& pt);
```

Implemente também a classe NaiveGuesser (adivinho ingénuo) que herda convenientemente da classe Guesser. Implemente o método

```
POINTTYPE guessTypeOfSinglePoint (CPoint& pt);
```

Este método deve adivinhar aleatoriamente (com probabilidade uniforme) o tipo do ponto pedido. Teste convenientemente esta classe.

b) Vamos agora começar a implementar a classe NeighbourGuesser (adivinho para o vizinho) que herda convenientemente da class Guesser.

Esta classe vai implementar um adivinho mais inteligente. Vai usar um conjunto de pontos cujo tipo conhecemos para tentar adivinhar o tipo de um ponto cujo tipo se desconhece.

Implemente o construtor

```
NeighbourGuesser(const char * filename);
```

que tem como parametro de entrada o nome do ficheiro com a informação conhecida. O ficheiro contém sucessivamente o X, Y e TIPO de cada ponto. Use este ficheiro para preencher os vectores

```
vector<CPoint> knownPoints;
```

```
vector<Guesser::POINTTYPE> knownTypes;
```

que deve acrescentar à sua classe NeighbourGuesser.

c) Implemente agora o membro

```
POINTTYPE guessTypeOfSinglePoint (CPoint& pt);
```

da classe `NeighbourGuesser`. Para prever o tipo para o ponto pedido este método deve procurar no vector `vector<CPoint> knownPoints` o ponto mais próximo e prever o tipo desse ponto (no vector `knownTypes`). Teste convenientemente esta função.



Programação 2

2º Semestre

Trabalho Prático P2B

Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Grupo 7

Considere a classe `BasicStat` que pretende oferecer estatísticas básicas (média e desvio padrão) de um conjunto de observações.

```
class BasicStat
{
    double sumX;
    double sumX2;
    unsigned count;

protected:
    double mean;
    double sdev;
    void addObservation (double x);

public:
    BasicStat () {count = 0, sumX = 0, sumX2 = 0;}
    virtual void addObservations (istream& is);
    virtual void printStat(ostream& os);
};
```

Note que para calcular estas estatísticas simples não é necessário acumular as observações. Basta ir actualizando a soma, `sumX`, e soma dos quadrados das observações, `sumX2`. No fim o

cálculo é simplesmente $media = \frac{\sum_{i=1}^{count} X_i}{count} = \frac{sumX}{count}$ e

$$desvio\ padrão = \sqrt{\left(\frac{\sum_{i=1}^{count} X_i^2 - count \times media^2}{count - 1}\right)} = \sqrt{\left(\frac{sumX2 - count \times media^2}{count - 1}\right)}$$

a) Implemente as funções

```
void addObservation (double x);
```

que actualiza os membros-dado de forma apropriada e

```
void addObservations (istream& is);
```

que lê todas as observações da entrada e usa a função anterior para actualizar os membros-dado.

Implemente ainda a função

```
virtual void printStat(ostream& os);
```

para escrever na stream de saída a média e desvio-padrão das observações lidas. Caso nenhuma observação tenha sido inserida deverá ser **lançada uma excepção apropriada**.

b) Pretende-se agora derivar da classe `BasicStat` a classe `AdvancedStat` que fornece informação adicional. Esta classe pretende ser capaz de fornecer os valores das observações subtraídos da média. Para tal, precisa de acumular todas as observações.

Implemente de forma conveniente as classe `AdvancedStat`, derivando da classe `BasicStat`.

A invocação da função

```
void printStat(ostream& os);
```

deve mostrar a média, o desvio padrão e também o valor das observações subtraídas da média.

c) Implemente a função

```
void lerSequenciasObservacoes (vector<BasicStat *>& seqObs);
```

que pede sucessivamente ao utilizador um nome de um ficheiro de observações e o tipo de estatística (“basic” ou “advanced”). Deve de seguida criar e inserir no vector `seqObs` um elemento apropriado contendo o tipo de estatística pretendido. A leitura deve ser terminada com **control-D**. Por fim teste a função no main e invoque o método `printStat` de todos os elementos inseridos no vector.



Programação 2

2º Semestre

Trabalho Prático P2B

Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Grupo 8

Pretende-se fazer um programa para codificar (encriptar) mensagens.

Considere a classe Coder:

```
class Coder
{
    virtual char codificaLetra(char c) = 0;
public:
    virtual string codificaMensagem(string& msg);
};
```

a) Implemente o membro-função de Coder

```
virtual string codificaMensagem(string& msg);
```

que codifica a mensagem dada como entrada. Esta função deve usar a função

```
virtual char codificaLetra(char c) = 0;
```

para codificar o mensagem.

Implemente também a classe BasicCoder (codificador básico) que herda convenientemente da classe Coder. Implemente o construtor

```
BasicCoder (int jump);
```

e o membro-função

```
char codificaLetra(char c);
```

Este codificador básico substitui cada letra pela letra `jump` vezes à frente no alfabeto. Por exemplo, se

`jump = 3`

a mensagem “Zebra” resulta em “Cheud”. Apenas as letras minúsculas e maiúsculas devem ser alteradas; os restantes caracteres não são alterados. Teste convenientemente esta classe.

b) Implemente agora a classe AdvancedCoder. Herde convenientemente da classe

BasicCoder. Esta classe deve ter o construtor

```
AdvancedCoder (string senha);
```

Deve voltar a implementar o membro-função

```
char codificaLetra(char c);
```

Nesta classe o avanço a fazer a cada letra depende da palavra senha. A senha contém sempre apenas letras minúsculas. Por exemplo `senha = "bife"`. A primeira letra a codificar deve ser avançada de 1 (‘b’-‘a’), a segunda letra a codificar deve avançar de 8 (‘i’-‘a’), a terceira de 5, a quarta de 4. A quinta letra a codificar volta a ser avançada de 1, etc. Assim a mensagem “Ola Ola” resulta em “Ptf Smi” (usando a senha “bife”).

c) Preencha convenientemente um vector

```
Coder * coders[2];
```

O primeiro elemento deve ser um apontador para um objecto BasicCoder e o segundo para um objecto do tipo AdvancedCoder. O salto para o primeiro objecto e a senha para o segundo objecto devem ser passados ao programa como argumentos (`argv[1]` e `argv[2]`).

Finalmente, peça sucessivamente ao utilizador uma frase (linha) e mostre a mensagem codificada por cada um dos Coder. O ciclo deve ser terminado com **control-D**.