



Programação 2

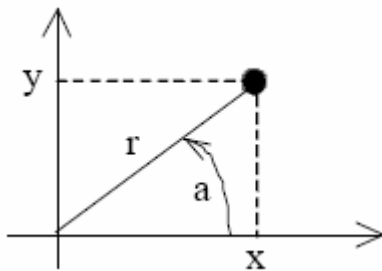
2º Semestre

Folha de Exercícios 1

Funções, argumentos, entrada/saída de dados

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

1. Pretendem-se escrever funções para converter coordenadas polares para rectangulares e vice-versa, de acordo com a figura e fórmulas indicadas de seguida.



$$y = r \sin a \quad (a \text{ em radianos})$$

$$x = r \cos a \quad (a \text{ em radianos})$$

$$r = \sqrt{x^2 + y^2}$$

$$a = \text{atan2}(y, x) \quad (a \text{ em radianos entre } -\pi \text{ e } \pi)$$

(Nota: *sin*, *cos*, *sqrt* e *atan2* são funções definidas em "math.h")

Cada uma destas funções deve ter dois parâmetros de entrada (as coordenadas a converter), passados por valor, e dois parâmetros de saída (as coordenadas resultantes da conversão), passados por referência, de acordo com os seguintes protótipos:

```
void polaresParaRectang(double r, double a, double & x, double & y);
void rectangParaPolares(double x, double y, double & r, double & a);
void rectangParaPolares(double x, double y, double * rPtr, double * aPtr);
```

Escreva um pequeno programa (main) para testar estas funções. Para usar as funções matemáticas é necessário incluir "math.h" ou compilar com "-lm".

2. Um número é primo se apenas for divisível pela unidade e por si próprio. Diz-se que um número a é divisível por um número b , se o resto da divisão inteira de a por b for zero.

a) Escreva uma função `bool primo(int n)` que retorna `true` se n é primo e `false` no caso contrário.

Sugestão: Para verificar se n é primo, basta dividir n pelos números de 2 a $\text{int}(\sqrt{n})$. A função `sqrt` retorna a raiz quadrada de um número.

b) Recorrendo à função anterior, escreva um programa que imprime os números primos compreendidos entre 1 e N , em que N é um valor indicado pelo utilizador.

c) Escreva uma versão do programa anterior em que N seja especificado na linha de comando.

Para poder aceder aos argumentos passados na linha de comando, é necessário declarar

```
int main(int argc, char *argv[])
```

em que `argc` é o número de argumentos (incluindo o nome do programa) e `argv` é o *array* de argumentos (strings do C).

Assim, ao executar `primos 120` vai acontecer que `argc` é 2, `argv[0]` é "primos" e `argv[1]` é "120".

3. Pretende-se escrever um programa denominado “meses” que disponibiliza informação genérica acerca dos meses do ano.

- a) O programa deve ler do *standard input* nomes de meses escritos exclusivamente com caracteres minúsculos, e escrever o número do mês, e sua abreviatura (três primeiros caracteres seguidos de um ponto). Se o nome do mês lido for inválido, o programa deve imprimir a mensagem “mês inválido”.

Experimentar o programa com entrada de dados do teclado. Para sinalizar o fim da entrada de dados a partir do teclado, deve digitar `ctrl-D`.

- b) A versão seguinte do programa deve ser chamada obrigatoriamente com um argumento que indica o número de caracteres a usar na escrita da abreviatura do mês. Por exemplo:

```
meses 5
```

significa que na escrita da abreviatura se devem usar os primeiros 5 caracteres. Se o mês tiver menos que 5 caracteres, a abreviatura é o nome completo.

Se a chamada for realizada sem argumentos, o programa deve imprimir uma mensagem de erro.

Nota: use a rotina `atoi()` para obter a representação numérica de uma cadeia de caracteres

- c) O programa pode ainda ser chamado com mais um argumento. Este argumento adicional é o nome do ficheiro que contém os nomes dos meses (em vez de ler do *standard input*). Por exemplo:

```
meses 5 meses.txt
```

Sugestão: Escrever uma função `processa_ficheiro` para tratar um *stream*, que pode ser um ficheiro já aberto ou o *standard input*. Esta função deve ter um argumento do tipo referência para `istream`. Note que onde se espera uma variável do tipo `istream` (como é o caso de `cin`) também se pode passar uma variável de um tipo derivado, como é o caso de `ifstream`.

4. Escreva um programa que aceita como parâmetros os nomes de vários ficheiros de texto e apresenta em *standard output* a concatenação do conteúdo dos ficheiros (i.e. comporta-se como o programa “cat” de Unix). Se não for indicado nenhum ficheiro na linha de comando, o programa deve afixar uma mensagem de erro.

5. Identifique e corrija os erros em cada uma das alíneas seguintes. (Nota: pode existir mais do que um erro por alínea; as variáveis não declaradas são inteiras)

- a)

```
if (x = y)
    cout << x << " é igual a " << y << "\n";
```

- b)

```
if (idade >= 65);
    cout << "A idade é maior ou igual que 65\n";
else
    cout << "A idade é menor que 65\n";
```

- c) O código seguinte deve imprimir se um dado inteiro (n) é par ou ímpar:

```
switch (n % 2) {
case 0:
    cout << "Inteiro par\n";
case 1:
    cout << "Inteiro ímpar\n";
```

```
}
```

d) O código seguinte deve imprimir os inteiros ímpares de 999 até 1:

```
for (int x = 999, x >= 1, x -= 2)
    cout << x << '\n';
```

6. Pretende-se escrever um programa denominado `opseq` para cálculo de medidas matemáticas.

a) O programa deve aceitar um número variável de números inteiros como argumentos, e apresentar a média desses números.

b) A nova versão do programa deve aceitar um argumento adicional (este novo argumento deve necessariamente ser o primeiro), o qual indica a operação a efectuar. Por exemplo:

```
opseq max 2 5 8 32 23
```

significa que o programa deve escrever no *standard output* o valor máximo da sequência de valores lidos.

As operações a implementar são:

- `max` (determina o máximo)
- `gama` (calcula a gama de valores = máximo-mínimo)
- `med` (calcula a média).