CADERNO DE EXERCÍCIOS 01

Biblioteca de *Streams* do C++
cout, cin

Em analogia ao C, o C++ define *standars* de *streams* de *input* e de *output*.
As streams são:

- *cout*, análogo ao *stdout*,
- *cin*, análogo ao *stdin*,

A entrada e a saída de dados com estas *streams* é efectuada recorrendo aos seguintes operadores:

- << Operador insersor
- >> Operador extractor

1.1 - Programa para ler o nome e a idade de uma pessoa.

```
#include <iostream>
    using namespace std;
    int main ()
    {
       int idade;
       char nome[30];

       cout << "Escreva o seu nome:" << endl;
       cin >> nome;
       cout << "Insira a sua idade:" << endl;
       cin >> idade;
       cout <<"O seu nome e: "<< nome <<" e tem: "<<idade<<"anos" << endl;
        system("pause");
       return (0);
    }
```

O programador não necessita de se preocupar com o tipo de dados que pretende utilizar, as *streams* adaptam-se em conformidade com a variável utilizada.

1.2 - Desenvolva um programa que permita fazer a leitura e escrita dos seguintes tipos de dados:

inteiro
real
caracter
string

Manipuladores em C++: endl, setw, setprecision, setf

O C++ disponibiliza vários manipuladores de *input/output* para formatação dos dados. Estes manipuladores mudam o formato das inserções.
endl – Manipulador usado para mudar de linha e limpar o *buffer*.
setw – Manipulador que define a largura do campo atribuído para o *output*. A largura do campo determina o número mínimo de caracteres a serem escritos na representação de saída.
setfill – Usa o caractere de preenchimento nas operações de inserção de *output* para preencher espaços quando os resultados devem ser preenchidos na largura do campo.
setprecision – Define o número total de dígitos a serem exibidos quando são impressos números de virgula flutuante.

1.3 - Exemplo do uso do manipulador *setw* e *setfill*

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
    {
       cout<<"ENSAIO do setw() ..............  " <<endl;
       cout<< setw(10) <<11<< endl;
       cout<< setw(10) <<2222<< endl;
       cout<< setw(10) <<4<< endl;
       cout<< setfill('-')<< setw(10) <<11<< endl;
       system("pause");
      return (0);
    }
```

1.4 - Exemplo do uso do manipulador *setprecision*

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
cout<<"Precisao global ....................... " << endl;
cout<<"5 digitos, parte inteira e decimal "<< setprecision(5) <<1234.537<< endl;
cout<<"6 digitos parte inteira e decimal "<<setprecision(6)<<1234.537<<endl<< endl;

cout<<"Precisao da parte decimal ...................... " << endl;
cout.setf(ios::floatfield,ios::fixed);
cout<< "2 digitos na parte decimal "<< setprecision(2)<<1234.537<< endl;
cout<< "5 digitos na parte decimal "<< setprecision(5)<<1234.537<< endl;

system("pause");
return 0;
}
```

Escrita e Leitura de Ficheiros

O exemplo seguinte lê dois valores inteiros guardados no ficheiro "dados_ent" e escreve no ficheiro "dados_saida" o quadrado destes valores. O primeiro código está escrito usando o paradigma tradicional (linguagem C) e o segundo usando objectos (C++).

2.1 - Exemplo de leitura e escrita em ficheiros em linguagem C

```
#include <stdio.h>
#include <stdlib.h>
using namespace std;
main()
{
    FILE *f1, *f2;
    int a, b;

    // abre o ficheiro dados_ent.txt no modo de leitura e associa-o a f1
    if( (f1 = fopen( "dados_ent.txt", "r" )) == NULL ){
         printf("ERRO: não é possível abrir o ficheiro dados_ent.txt\n");
         exit(1);
    }

    // abre o ficheiro dados_saida.txt no modo de escrita e associa-o a f2
    if( (f2 = fopen( "dados_saida.txt", "w" ))== NULL ){
         printf("ERRO: não é possível abrir o ficheiro dados_saida.txt\n");
         exit(1);
    }

    fscanf( f1, "%d", &a ); // leitura e escrita
    fscanf( f1, "%d", &b );

    printf("%d %d\n",a,b);

    fprintf( f2, "Este ficheiro foi alterado na aula N°  .\n" );
    fprintf( f2, "%d\n", a*a );
    fprintf( f2, "%d\n", b*b );

    fclose( f1 );  // fecha o ficheiro associado a f1
    fclose( f2 );  // fecha o ficheiro associado a f2
    system("pause");
    return 0;
}
```

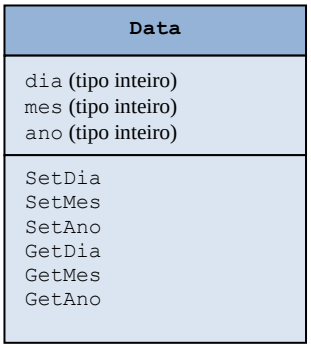
2.2 - Exemplo de escrita e leitura de ficheiros na linguagem C++: Classes **ifstream** e **ofstream**

```
#include <fstream >
#include <iostream>
using namespace std;
int main()
{
    ifstream is; // objecto os para abrir o ficheiro em modo de leitura
    ofstream os; // objecto os para abrir o ficheiro em modo de escrita
                 // para abrir o ficheiro em modo de escrita e leitura
                  // deve usar-se a class fstream

    is.open("dados_ent.txt");
    if( !is ){
        cout << "ERRO: não é possível abrir o ficheiro dados_ent.txt" << '\n';
        exit(1);
    }

    os.open("dados_saida.txt");
    if( !os ){
      cout << "ERRO: não é possível abrir o ficheiro dados_saida.txt" << '\n';
      exit(1);
    }

    int a, b;

    is >> a >> b;

    cout << a << endl << b << endl;

    os << a*a << ' ' << b*b;

    is.close();  // fecha o ficheiro associado a is
    os.close();  // fecha o ficheiro associado a os
    system("pause");
    return 0;
}
```

2.3 - Desenvolva um programa em C++ que leia do teclado para uma estrutura a informação de 3 alunos (Nome, Idade e Número) e de seguida grave no ficheiro "***alunos.txt***" a informação de cada aluno numa linha, usando o separador ";" entre os campos.

2.4 - Faça um programa que leia vários campos de dados (Nome, Idade e Número), separados por "ponto e vírgula", guardados no ficheiro "***alunos.txt***" e os apresente no ecrã.

3. Crie a classe `Data` com os atributos e métodos a seguir apresentados.



a) Implemente dois construtores para a classe `Data`. Um construtor por defeito (sem parâmetros) e um construtor com parâmetros definidos pelo utilizador.
b) Implemente todos os métodos `Get()` e `Set()` de maneira a que sejam métodos *inline*.

Métodos *inline*

Tem as características normais dos métodos (sintaxe, verificação dos argumentos, etc), mas são expandidos em *compile-time* em vez de invocados em *run-time*.

O código destes métodos deve estar no ficheiro de especificação (.h), pelo que a sua alteração implica a recompilação.

A declaração *inline* justifica-se apenas para métodos simples.

c) Implemente o método `Show()` para escrever os atributos dos objetos no ecrã.
d) Implemente o método `Update()` que permita actualizar todos os atributos de um objecto `Data`.

e) Faça um programa que:
- Crie 2 objetos do tipo `Data` usando para um o construtor por defeito e para outro o construtor por parâmetros.
- Invoque o método `Show()` usando cada um dos objetos anteriormente criados.
- Altere os atributos do objeto criado com o construtor por defeito com dados introduzidos pelo utilizador através do teclado.
- Altere os atributos do objeto criado com o construtor por parâmetros usando o método `Update()`.

f) Implemente para a classe `Data` o método `Igual()` para verificar se dois objectos do tipo `Data` são iguais. O método deve devolver `true` se forem iguais e `false` caso contrário.
g) Verifique se dois objectos do tipo `Data` são iguais fazendo a sobrecarga do operador "==".

Sobrecarga de operadores

Sobrecarga de operadores aritméticos: operador "+" e operador "−"

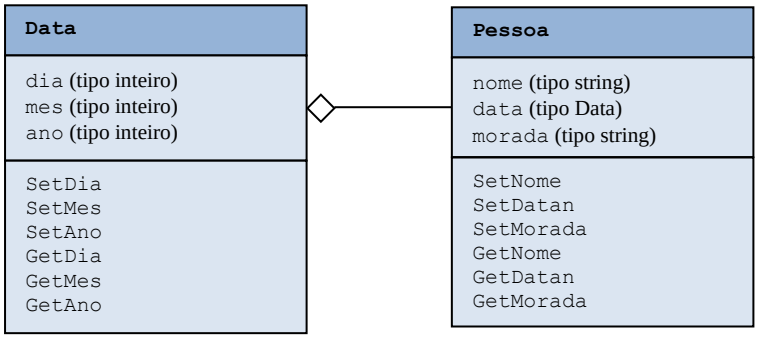
Sobrecarga de operadores relacionais: operador "!=" e operador "=="

Sobrecarga de operadores de entrada e saída de dados: operator "<<" e operator ">>"

h) Implemente a sobrecarga dos operadores "!=", "<<" e ">>".

i) Crie um novo objecto *Data* e leia os seus dados através do teclado usando o operador criado na alínea anterior.
j) Mostre no ecrã todos os objectos *Data* criados recorrendo ao operador "<<".
k) Implemente na classe Data métodos para leitura (*ReadFile*) e escrita (*SaveFile*) em ficheiro. Em seguida invoque estes métodos no seu programa.

4. Crie a classe `Pessoa` com os atributos e métodos a seguir apresentados utilizando funções *inline* para os métodos de acesso.



a) Implemente os construtores por defeito e com parâmetros da classe `Pessoa`.
b) Crie 2 objetos do tipo `Pessoa` com os seguintes atributos:
    a. Jonas Culatra; 20/9/1987; Rua da direita n 2.
    b. Joni Rato; 4/2/1990; Rua da esquerda n 3.
c) Implemente o método *Show()* e mostre no ecrã os dados das pessoas criadas na alínea anterior.
d) Altere a morada do "Joni Rato". A nova morada deve ser lida através do teclado.
e) Altere a data de nascimento do "Joni Rato". A nova data de nascimento deve ser lida através do teclado.
f) Implemente o método `ReadK()` que permita ler todos os dados de uma pessoa através do teclado. Crie um 3º objeto do tipo `Pessoa` e use este método para ler os seus dados.
g) Implemente a sobrecarga dos operadores "<<" e ">> " na classe `Pessoa`. Use o construtor por defeito para definir uma nova `Pessoa` e use os operadores "<<" e ">> " para ler e mostrar os seus dados.
h) Implemente a sobrecarga dos operadores "==", "!=", que faz a comparação do nome e data de nascimento; No programa verifique se 2 pessoas são iguais
i) Implemente o método `MaisNovo()` para verificar entre duas pessoas qual é a mais nova. Deve ser apresentado no ecrã a informação completa da Pessoa mais nova.
j) Implemente o método `SaveFile()` para guardar todos os dados de uma pessoa em ficheiro. Use este método para guardar os dados de todas as pessoas em ficheiro. O nome do ficheiro deve ser lido através do teclado.
k) Implemente o método `ReadFile()` que permita ler todos os dados de uma pessoa a partir de um ficheiro.
l) Implemente o código necessário para ler os dados do ficheiro, criado na alínea anterior, para um vetor de objectos do tipo `Pessoa`.
m) Apresente a informação completa das pessoas cuja data de nascimento seja anterior a 1990.