

C++ Multiplataforma e Orientação a Objetos

Exercícios da parte 6:

C++ e programação genérica (template)

Por: Sergio Barbosa Villas-Boas (sbVB)



<http://www.sbVB.com.br/c++.html>

Versão 8.0, de 06 de Outubro de 2003

Copyright © 1992~2003 by sbVB

Email do autor: sbvb@sbvb.com.br

Exercício 1)

Seja o arquivo de dados em formato texto com nome “parts.dat”, como mostrado abaixo. O formato dos dados é da seguinte forma: código da peça 1 (1001), peso 1 (20.5), comprimento 1 (10.25), preço 1 (100.20), código da peça 2 (1002), peso 2 (120.0), comprimento 2 (1.8), preço 2 (1100.00), etc. Por exemplo o arquivo seria como abaixo:

```
1001,20.5,10.25,100.20
1002,120.0,1.8,1100.00
etc
```

Complete o programa C++ abaixo, nos 8 trechos marcados com “add code here” para que execute a seguinte funcionalidade:

- Ler todas as peças do disco para a memória, numa lista encadeada.
- Alocar um array de n (número de peças) posições, copiar a lista encadeada para o array e liberar a lista encadeada.
- Perguntar ao usuário se deseja ordenar a peça por peso, comprimento ou preço.
- Listar as peças segundo a ordenação solicitada.

```
#include <iostream>
#include <fstream>
#include <stdlib.h> // exit
#include "vplib.h" // VList
using namespace std;

template <class T>
void bubble_sort(T* array, int n, bool (*compare)(T,T)) {
    for (int i=0 ; i < n ; i++)
        for (int j=i+1 ; j < n ; j++)
            if (compare(array[i],array[j]))
                // if (array[i] < array[j])
                { // swap (array[i], array[j])
                    T temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
}

class Part
{
    int m_partCode;
    double m_weight;
    double m_length;
    double m_price;
public:

    friend ostream & operator<<(ostream & stream, Part & obj)
    {
        // add code here (1)
    };
};
```

```
friend istream & operator>>(istream & stream, Part & obj)
{
    // add code here (2)
};

friend bool comparePartcode(const Part & a, const Part & b)
{
    // add code here (3)
}

friend bool compareWeight(const Part & a, const Part & b)
{
    // add code here (4)
}

friend bool compareLength(const Part & a, const Part & b)
{
    // add code here (5)
}

friend bool comparePrice(const Part & a, const Part & b)
{
    // add code here (6)
}
};

void printArray(const Part *partArray,int count)
{
    for (int i=0 ; i < count ; i++)
    {
        cout << "-----" << endl << partArray[i] << endl;
    }
}

int main()
{
    const char *fileName = "parst.dat";
    ifstream myFile(fileName);
    if (!myFile) {
        cout << "Could not open file" << endl;
        exit(1);
    }

    VBList<Part> myPartList;
    Part part, *p;

    int count = 0;
    // add code here (7)
    // the code to be added should read completely the file
    // and save all elements to linked list myPartList
    cout << "The file contains " << count << " parts" << endl;

    Part *partArray = new Part [count];

    // transfer list to array
    int i=0;
    for (p = myPartList.GetFirst() ; p ; p = myPartList.GetNext() )
    {
        partArray[i] = *p;
        cout << i++ << endl;
    }
}
```

```
cout << "Array before order" << endl;
printArray(partArray, count);

int choice;
// add code here (8)

switch (choice)
{
case 1:
    bubble_sort(partArray, count, comparePartcode);
    break;
case 2:
    bubble_sort(partArray, count, compareWeight);
    break;
case 3:
    bubble_sort(partArray, count, compareLength);
    break;
case 4:
    bubble_sort(partArray, count, comparePrice);
    break;
}

cout << "Ordered array" << endl;
printArray(partArray, count);

delete [] partArray;
return 0;
}
```

Exercício 2)

O programa abaixo usa a classe genérica de complex do C++ padrão. O número complexo possui dois números reais em ponto flutuante. Qual tipo ponto flutuante que se usa para definir os dois reais do número complexo é o parâmetro genérico da classe complex. No caso abaixo, usou-se double.

- Mostre o que o programa abaixo faz aparecer no console. (atenção: a multiplicação de um número complexo com outro número complexo não é a multiplicação das partes reais e imaginárias isoladamente).
- Escreva um trecho programa (acrescentado ao final da função main) para calcular o produto dos números complexos $(3+4i)$ com $(5+6i)$.

```
#include <iostream>
#include <complex>
using namespace std;

// define the T d_complex, that is,
// complex of 2 double elements (real and imag)
typedef complex<double> d_complex;

template<class T, unsigned size>
class MyVector
{
    T m_p[size];
public:
    void show()
```

```
{
    for (int i=0 ; i < size ;i++)
        cout << m_p[i] << endl;
};
MyVector (T data=0.0)
{
    for (int i=0 ; i < size ;i++)
        m_p[i] = data;
};
T & operator[](int i)
{
    T & ret = m_p[i];
    return ret;
}
MyVector operator+(MyVector & a)
{
    MyVector ret;
    for (int i=0 ; i < size ;i++)
        ret[i] = (*this)[i] + a[i];
    return ret;
}
void operator=(MyVector & a)
{
    for (int i=0 ; i < size ;i++)
        (*this)[i] = a[i];
}
friend ostream & operator<< (ostream & s, MyVector & a)
{
    for (int i=0 ; i < size-1 ;i++)
        s << a[i] << " ";
    s << a[i];
    return s;
}
}; // end of class MyVector

int main()
{
    // 3 vectors of double
    const int size = 3;
    MyVector<double,3> d_m, d_n, d_k;
    int i;
    for (i=0 ; i < size ; i++) {
        d_m[i] = i+1;
        d_n[i] = 2*i-1;
    }
    cout << d_m << endl;
    cout << d_n << endl;
    d_k = d_m + d_n;
    cout << d_k << endl;

    // 3 vectors of complex
    MyVector<d_complex,size> c_m, c_n, c_k;
    for (i=0 ; i < size ; i++)
    {
        c_m[i] = d_complex(i+1,i+2); // real, imag
        c_n[i] = d_complex(2*i-1,3*i); // real, imag
    }
    cout << c_m << endl;
    cout << c_n << endl;
    c_k = c_m + c_n;
    cout << c_k << endl;
}
```

```
    return 0;
}
```

Exercício 3)

O programa abaixo usa a classe de complex com template padrão do C++. Para usar essa classe, é preciso usar namespace. A partir da definição de complex com template, cria-se um tipo complexo do usuário, usando-se o tipo de ponto flutuante que se queira (no caso desse problema usou-se double).

- Mostre o que o programa abaixo faz aparecer no console. (atenção: a multiplicação de um número complexo com outro número complexo não é a multiplicação das partes reais e imaginárias isoladamente).

```
#include <iostream>
#include <complex>
using namespace std;

template<class T>
class MyVector
{
    T *m_p;
    int m_size;
public:

    // default constructor
    MyVector(int size=0)
    {
        m_size = size;
        if (m_size > 0)
            m_p = new T [m_size];
    }

    ~MyVector()
    {
        if (m_p)
        {
            delete [] m_p;
            m_p = 0;
        }
    }

    void show()
    {
        for (int i=0 ; i < m_size ;i++)
            cout << m_p[i] << endl;
    };

    T & operator[](int i)
    {
        T & ret = m_p[i];
        return ret;
    }

    MyVector operator+(MyVector & a)
    {
        MyVector ret;
        for (int i=0 ; i < m_size ;i++)
```

```
        ret[i] = (*this)[i] + a[i];
    return ret;
}

void operator=(MyVector & a)
{
    for (int i=0 ; i < m_size ;i++)
        (*this)[i] = a[i];
}

friend ostream & operator<< (ostream & s, MyVector & a)
{
    for (int i=0 ; i < a.m_size ;i++)
        s << a[i] << "\t";
    return s;
}
};

typedef complex<double> d_complex;
typedef MyVector<d_complex> MyVectorComplex;

int main()
{
    int size = 4;
    MyVectorComplex v(size);

    for (int i=0 ; i < size ; i++)
    {
        v[i] = d_complex(1.1+i, 1.5+2*i);
    }

    cout << v[3]*v[1] << endl;
    return 0;
}
```

Exercício 4)

Diga se o programa abaixo contém um bug ou não. Explique porque.

Altere o método que suporta o `operator()`, da classe `myArray`, para que exista tratamento de *range-check* dentro da classe `myArray`. O tratamento de *range-check* significa que se o parâmetro do `operator()` for fora do limite existente, a classe acusa erro, e não tenta acessar o elemento inexistente do array.

```
#include "vplib.h"

template <class T>
class MyArray
{
    T* m_data;
    int m_size;
public:
    MyArray(int size=0)
    {
        m_size = size;
        m_data = new T [size];
    }

    ~MyArray()
```

```
{
    if (m_data)
        delete [] m_data;
}

T & operator()(int i)
{
    return m_data[i];
}
};

struct TestClass
{
    VBString m_portuguese;
    VBString m_english;
};

int main()
{
    MyArray<double> a(4);
    a(0) = 0.2;
    a(1) = 1.2;
    a(2) = 2.2;
    a(3) = 3.2;
    a(4) = 3.2;

    MyArray<testClass> b(10);
    b(0).m_portuguese = "zero";
    b(0).m_english   = "zero";
    b(1).m_portuguese = "um";
    b(1).m_english   = "one";
    b(2).m_portuguese = "dois";
    b(2).m_english   = "two";
    b(3).m_portuguese = "tres";
    b(3).m_english   = "three";
    b(4).m_portuguese = "quatro";
    b(4).m_english   = "four";

    return 0;
}
```

Exercício 5)

Deseja-se fazer um programa, com 2 nomes de arquivo como parâmetro - file_text e file_data, com a seguinte funcionalidade. O arquivo file_data contém pares de palavras no formato texto. Cada linha desse arquivo contém uma “palavra da esquerda” e uma “palavra da direita”, separado por vírgula. O programa deverá retornar no console (tela) o conteúdo de file_text, sendo que todas “palavras da direita” deverão estar trocadas pelas “palavras da esquerda”.

Exemplo do arquivo file_data:

```
morango,abacaxi
abacate,melancia
uva,goiaba
```

Exemplo do arquivo file_text:

Esse arquivo pode conter qualquer conteúdo na forma de texto.

Eu gosto de morango, abacaxi, abacate, melancia, uva e goiaba.
Qualquer outra coisa.

Exemplo do arquivo file_text após execução do programa:

Esse arquivo pode conter qualquer conteúdo na forma de texto.
Eu gosto de abacaxi, abacaxi, melancia, melancia, goiaba e goiaba.
Qualquer outra coisa.

Para fazer esse programa, complete o programa abaixo. Dica: primeiro faça o programa copiar o texto para a tela sem nenhuma troca. Depois experimente o método replace da VBString e troque alguma palavra manualmente [e.g. str.replace("morango","abacaxi")]. Em seguida, leia todo o conteúdo do arquivo file_data para uma lista encadeada, com tantos elementos quantas forem os pares de palavras em file_dat. Por fim, faça um loop de trocas, chamando o método replace tantas vezes quantas forem os elementos da lista encadeada.

```
#include <fstream>
#include "vplib.h"

struct StringPairToChange
{
    VBString m_search;
    VBString m_replace;
};

typedef VBList<StringPairToChange> ListToChange;

void do_all (const char *fileChangeData, const char* fileMain)
{
    StringPairToChange data; // one instance of data
    StringPairToChange *p; // pointer to one instance of data
    ListToChange mylist; // instance of list
    VBString str;
    // add code here
}

int main()
{
    do_all("file_data.txt","file_main.txt");
    return 0;
}
```