НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Програмування ч.2 »

*Звіт з лабораторної роботи №11*

*Тема:* Шаблонні класи

Виконав:

ст. гр. KIT-119D

Гряник Г.В.

Перевірив:

Асистент Віктор ЧЕЛАК

Асистент Владислав ЯЛОВЕГА

Харків – 2020

***Мета:*** Пошири знання у шаблонізації (узагальненні) на основі  
вивчення шаблонних класів та створення власних шаблонних типів.

Варіант 6.

Загальне завдання.Модернізувати клас, що був розроблений у попередній роботи  
наступним шляхом:  
− зробити його шаблонним;  
− додати поле – шаблонний масив;  
− видалити з аргументів існуючих методів масив, а замість цього  
використовувати масив-поле класу.  
Необхідно продемонструвати роботу програми як з використанням  
стандартних типів даних, так і типів, які створені користувачем.

**індивідуальні завдання**

Тексt програми

prototaype.h

……….

without\_fields.h

template <class T >

class List: public VList<T>

{

T\* data[ARRAY\_SIZE];

int index = -1;

public:

void add(T\* el)

{

if (index + 1 >= ARRAY\_SIZE)return;

else this->data[++index] = el;

}

void show() override

{

cout << "Ваш масив :"<<endl;

for (int i = 0; i <= index; i++)

data[i]->show();

cout << endl;

}

void print\_min\_Element()

{

cout << "Мінімальний елемент масиву: ";

T\* a =data [0];

for (int i = 0; i < index+1; i++)

a = GetMin(a, data[i]);

a->show();

cout << endl << endl;

}

T\* GetMin(T\* aLeft, T\* aRight)

{

return aLeft < aRight ? aLeft : aRight;

}

void sortEL()

{

T\* temp;

bool pr; // для проверки отсортирован ли массив

do

{

pr = false;

for (int i = 0; i < index; i++) // проходим слева направо

if (data[i] > data[i + 1])

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

pr = true;

}

} while (pr);

cout << " \*сортування завершено " << endl;

}

void poisk()

{

int poshuk;

cout << "Яке число хочете знайти :" << endl;

cin >> poshuk;

cout << endl;

int a = 0;

for (int i = 0; i < index; i++)

if ( poshuk==data[i])

{

cout << ++a << "Індекс вашого елементу: " << i << endl;

}

if (a == 0)

cout << "Такого елементу немає в масиві" << endl;

}

void del()

{

while (index - 1 < -1)

{

index--;

}

}

T\* Get\_Elem(int Index) { return data[Index]->clone(); }

//////

T& operator=(const T& rhs) {

if (this == &rhs) // a = a;

return \*this;

while (index >= 0)

delete data[index--];

for (int i = 0; i < rhs.index; i++) {

data[i] = rhs.data[i]->clone();

}

index = rhs.index;

}

T\* operator [](const int index) { return data[index]->clone(); }

};

class IAB {

public:

virtual void show() = 0;

virtual IAB\* clone() = 0;

};

class fields:public IAB

{

private:

int x;

int y;

public:

fields()

{

x = 0;

y = 0;

}

fields(int x, int y)

{

this->x = x;

this->y = y;

}

~fields() { ;}

int getX() { return x; }

int getY() { return y; }

void setX(int ax) { x = ax; }

void setY(int ay) { y =ay; }

IAB\* clone() {return (IAB\*)new fields(\*this);}

void show()

{ cout << "x=" << getX() << "; y=" << getY() << ";" <<endl;

}

friend bool operator < (fields p, fields p2) {

int i = 0;

return (p.getX() + p.getY() < p2.getX() + p2.getY());

}

friend bool operator > (fields p, fields p2) {

return (p.getX() + p.getY() > p2.getX() + p2.getY() );

}

friend ostream& operator << (ostream& out, fields& pr)

{

return out << "x=" << pr.getX() << "; y=" << pr.getY() << ";";

}

friend bool operator== (int p, fields& p2)

{

bool a = false;

if (p == p2.getX()) a = true;

else if (p == p2.getY()) a = true;

return a; }

friend ifstream& operator >> (ifstream& in, fields& pr)

{

int x;

in >> x;

pr.setY(x);

}

fields& operator = (const fields& SR)

{

// cout << "Перегрузка оператора '='" << endl;

if (this == &SR)

return \*this;

x = SR.x;

y = SR.y;

return \*this;

}

};

class fields2:public fields

{

private:

int z;

public:

fields2()

{

z = 0;

}

fields2(int x,int y,int z)

{

setX(x);

setY(y);

this->z = z;

}

~fields2() { ; }

IAB\* clone() { return (IAB\*)new fields2(\*this); }

int getZ() { return z; }

void setZ(int Z) { z = Z; }

void show()

{

cout << "x=" << getX() << "; y=" << getY() << "; z=" << getZ() << ";" << endl;

}

friend bool operator < (fields2 p, fields2 p2) {

int i = 0;

return (p.getX() + p.getY() + p.getZ() < p2.getX() + p2.getY() + p2.getZ());

}

friend bool operator > (fields2 p, fields2 p2) {

return (p.getX() + p.getY() + p.getZ() > p2.getX() + p2.getY() + p2.getZ());

}

friend ostream& operator << (ostream& out, fields2& pr)

{

return out << "x=" << pr.getX() << "; y=" << pr.getY() << "; z=" << pr.getZ() << "; t=" << pr.getZ() << ";" ;

}

friend bool operator== (fields2& p, fields2& p2) { return !(p < p2) && !(p > p2); }

friend bool operator== (int p, fields2& p2)

{

bool a = false;

if (p == p2.getX()) a = true;

else if (p == p2.getY()) a = true;

else if (p == p2.getZ())a = true;

return a;

}

};

class fields3 :public fields2

{

private:

int t;

public:

fields3()

{

t = 0;

}

fields3(int x, int y, int z, int t)

{

setX (x);

setY (y);

setZ (z);

this->t = t;

}

~fields3() { ; }

IAB\* clone() { return (IAB\*)new fields3(\*this); }

int getT() { return t; }

void setT(int T) { t = T; }

void show()

{

cout << "X" << getX() << "; Y=" << getY() << "; Z=" << getZ() << "; T=" << getT() << ";" << endl;

}

friend bool operator < (fields3 p, fields3 p2) {

return (p.getX() + p.getY() + p.getZ() + p.getT() < p2.getX() + p2.getY() + p2.getZ() + p2.getT());

}

friend bool operator > (fields3 p, fields3 p2) {

return (p.getX() + p.getY() + p.getZ() + p.getT() > p2.getX() + p2.getY() + p2.getZ() + p2.getT());

}

friend ostream& operator << (ostream& out, fields3& pr)

{

return out << "x=" << pr.getX() << "; y=" << pr.getY() << "; z=" << pr.getZ() << "; t=" << pr.getT() << ";";

}

friend bool operator== (fields3& p, fields3& p2) { return !(p < p2) && !(p > p2); }

friend bool operator== (int p, fields3& p2)

{

bool a = false;

if (p == p2.getX()) a = true;

else if (p == p2.getY()) a = true;

else if (p == p2.getZ())a = true;

else if (p == p2.getT())a = true;

return a;

}

};

class fields4 :public fields3

{

private:

int p;

public:

fields4()

{

p = 0;

}

fields4(int x, int y, int z, int t,int p)

{

setX(x);

setY(y);

setZ(z);

setT(t);

this->p =p ;

}

~fields4() { ; }

IAB\* clone() { return (IAB\*)new fields4(\*this); }

int getP() { return p; }

void setP(int P) { p = P; }

void show()

{

cout << "X" << getX() << "; Y=" << getY() << "; Z=" << getZ() << "; T=" << getT() << "; P=" << getP() << ";" << endl;

}

friend bool operator < (fields4 p, fields4 p2) {

return (p.getX() + p.getY() + p.getZ() + p.getT() < p2.getX() + p2.getY() + p2.getZ() + p2.getT());

}

friend bool operator > (fields4 p, fields4 p2) {

return (p.getX() + p.getY() + p.getZ() + p.getT() > p2.getX() + p2.getY() + p2.getZ() + p2.getT());

}

friend ostream& operator << (ostream& out, fields4& pr)

{

out << "X" << pr.getX() << "; Y=" << pr.getY() << "; Z=" << pr.getZ() << "; T=" << pr.getT() <<"; P=" << pr.getP() << ";"<<endl;

}

friend bool operator== (fields4& p, fields4& p2) { return !(p < p2) && !(p > p2); }

friend bool operator== (int p, fields4& p2)

{

bool a = false;

if (p == p2.getX()) a = true;

else if (p == p2.getY()) a = true;

else if (p == p2.getZ()) a = true;

else if (p == p2.getT()) a = true;

return a;

}

};

Main.cpp

#include "prototaype.h"

#include "without\_fields.h"

typedef int real;

typedef double vegas;

int main()

{

cout << " шаблон " << endl;

fields d(1, 11);

fields d4(4, 1);

fields d6(6, 6);

List<fields> C;

C.add(&d);

C.add(&d);

C.add(&d4);

C.add(&d);

C.show();

C.print\_min\_Element();

C.sortEL();

C.show();

system("pause&cls");

List<IAB> D;

cout << " шаблон із спадкоємцями " << endl;

fields4 d2(2, 4, 133,6,43) ;

fields2 d5(5, 1, 33);

fields3 d3(3, 31, 13, 15);

fields3 d7(7, 1, 3, 7);

for (int i = 0; i < 4; i++)

{

D.add(&d);

D.add(&d2);

D.add(&d3);

D.add(&d4);

D.add(&d5);

D.add(&d6);

D.add(&d7);

}

D.show();

D.print\_min\_Element();

D.sortEL();

D.show();

//D.poisk();

return 0;

}

**Результати роботи програми:**

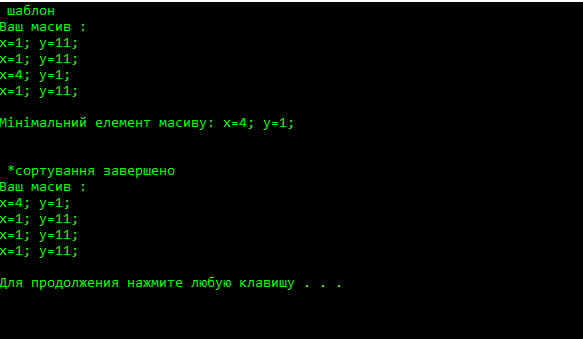


Рисунок 1 – Результати роботи програми шаблоного класу List

**Висновки:**

В результаті виконання лабораторної роботи отримав знання у шаблонізації на основі вивчення шаблонних класів та створення власних шаблонних типів.