МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Харківський політехнічний інститут»

Факультет Комп’ютерних наук та програмної інженерії

Кафедра Інформатики та інтелектуальної власності

ЗВІТ

До лабораторной роботи №4 з дисципліни

«Об’єкто-орієнтоване програмування»

Студент Бородай Д.А

Викладач Івашко А.В.

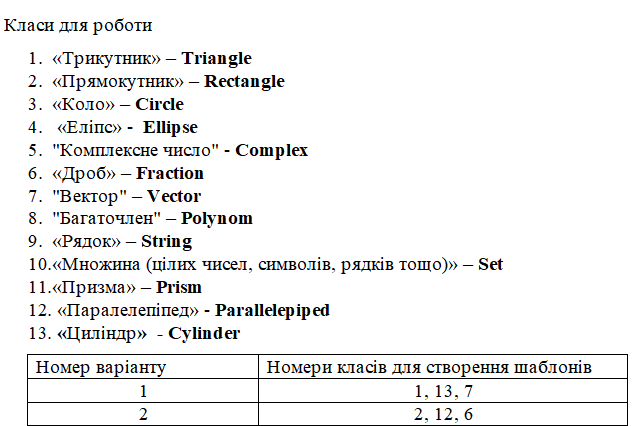
Харків 2022

**Лабораторна робота №4**

**Тема: «Шаблони класів та шаблони функцій»**

**Мета роботи:** навчитись реалізовувати та використовувати шаблони функцій та шаблони класів, визначити переваги та недоліки кожного шаблону.

**Завдання:**



Для класів, заданих за власним варіантом, перевантаживши відповідні оператори створити нижче вказані шаблони.

1. Шаблони функцій пошуку з трьома аргументами, якими є об’єкти відповідних класів:

· максимального значення,

· максимального значення,

· середнього арифметичного значення,

· середнього квадратичного значення,

· суми аргументів,

· добуток чисел.

2. Шаблони функцій пошуку, аргументами яких є масиви об’єктів відповідних класів

· максимального елементу масиву,

· мінімального елементу масиву,

· суму елементів масиву,

· добуток елементів масиву,

· середнє арифметичне масиву,

· середнє арифметичне масиву,

· кількість елементів масиву, що відповідають певному (створити самостійно) критерію.

3. Об’єднати функції першого завдання в клас для якого створити відповідний шаблон.

4. Об’єднати функції другого завдання в клас для якого створити відповідний шаблон.

Зміст

[1 Аналіз до предметної області 5](#_Toc119007156)

[2 Виконання роботи 6](#_Toc119007157)

[3 Довідка по роботі з програмою для користувача. 9](#_Toc119007158)

[Висновок 10](#_Toc119007159)

## 1 Аналіз до предметної області

Часто, при розробці класів для різних типів даних, програмісту доводиться писати програмний код для кожного типу окремо. Методи та операції над даними різних типів можуть містити той самий повторюваний код. Щоб уникнути повторюваності написання коду для різних типів даних, у мові C++ використовуються так звані шаблони.

Шаблон класу дозволяє оперувати даними різних типів у загальному. Тобто, немає прив’язки до деякого конкретного типу даних (int, float, …). Вся робота виконується над деяким узагальненим типом даних, наприклад типом з іменем T.

Фактично, оголошення шаблону класу є тільки описом. Створення реального класу з заданим типом даних здійснюється компілятором в момент компіляції, коли оголошується об’єкт класу.

## 2 Виконання роботи

Для опрацювання з задачею було створено два класи Task1 та Task2, які приймають відповідно об’єкти класу або масив об’єктів. Згідно вимог до задач, було створено відповідні методи. Приклад реалізації таких класів наведено нижче.

template <class T>

class Task1

{

T ob1, ob2, ob3;

public:

Task1(){}

Task1(T a, T b, T c)

{

this->ob1 = a;

this->ob2 = b;

this->ob3 = c;

}

void maxim()

{

T arry[3] = { ob1, ob2, ob3 };

T max = arry[0];

for (int j = 0; j < 3; j++)

if (arry[j] > arry[j - 1])

max = arry[j];

cout << "The maximal is: " << max << endl;

}

void minim()

{

T arry[3] = { ob1, ob2, ob3 };

T min = arry[0];

for (int j = 0; j < 3; j++)

if (arry[j] < arry[j - 1])

min = arry[j];

cout << "The minimal is: " << min << endl;

}

void arith\_mean()

{

T temp = ob1 + ob2;

temp = temp + ob3;

temp = temp / 3;

cout << "Arithmetical mean: " << temp << endl;

}

void sqrt\_mean()

{

T t1 = ob1 \* ob1;

T t2 = ob2 \* ob2;

T t3 = ob3 \* ob3;

T t4 = t1 + t2;

T t5 = t4 + t3;

T temp = t5 / 3;

temp.squrt();

cout << "Medium-square mean: " << temp << endl;

}

void summary()

{

T t1 = ob1 + ob2;

t1 = t1 + ob3;

cout << "The summary is: " << t1 << endl;

}

void multipliance()

{

T t1 = ob1 \* ob2;

t1 = t1 \* ob3;

cout << "The multipliance is: " << t1 << endl;

}

};

template <class T>

class Task2

{

int size;

T\* ob;

public:

Task2()

{

this->ob = nullptr;

this->size = 0;

}

Task2(T\* obj, int size)

{

this->size = size;

this->ob = obj;

}

void maxim()

{

T max = ob[0];

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

{

if (max <= ob[i])

max = ob[i];

}

cout << "The maximal element:\n";

cout << max << endl;

}

void minim()

{

T min = ob[0];

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

{

if (min >= ob[i])

min = ob[i];

}

cout << "The minimal element:\n";

cout << min << endl;

}

void sum()

{

T sum = ob[0];

for (int i = 1; i < size; i++)

sum = sum + ob[i];

cout << "The summary is:\n";

cout << sum << endl;

}

void mult()

{

T mult = ob[0];

for (int i = 1; i < size; i++)

mult = mult \* ob[i];

cout << "The multipliance is:\n";

cout << mult << endl;

}

void mean()

{

T sum = ob[0];

for (int i = 1; i < size; i++)

sum = sum + ob[i];

sum = sum / this->size;

cout << "The mean is:\n";

cout << sum << endl;

}

void sq\_mean()

{

T sum = ob[0] \* ob[0];

for (int i = 1; i < size; i++){

T mult = ob[i] \* ob[i];

sum = sum + mult;

}

sum = sum / this->size;

sum.squrt();

cout << "The square mean is:\n";

cout << sum << endl;

}

void filter(T criteria)

{

int cnt = 0;

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

if (criteria == ob[i]) {

cnt++;

cout << i + 1 << ") " << ob[i] << endl;

}

if (cnt != 0)

cout << "There are " << cnt << " coincidenses\n";

else

cout << "Coincidenses hasn`t found!\n";

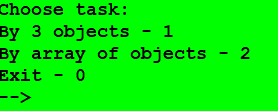
}

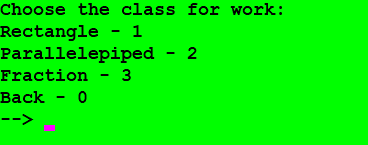
};

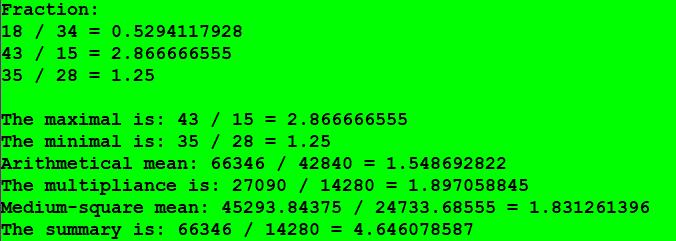
Повний код програми наведено у додатку А.

## 3 Довідка по роботі з програмою для користувача.

У головному меню користувач може обрати роботу з трьома об’єктами або з масивом об’єктів, розмір котрого вирішує задати користувач. У меню третього рівня користувач може обрати один з трьох класів та вивести відповідну інформацію. Приклади меню продемонстровано на рисунку 3.1, 3.2, 3.3.







## Висновок

Під час виконання лабораторної роботи було засвоєно на практиці користування шаблонними класами.

Оголошення шаблону класу дає такі переваги:

* уникнення повторюваності написання програмного коду для різних типів даних. Програмний код (методи, функції) пишеться для деякого узагальненого типу T;
* зменшення текстової частини програмного коду, і, як наслідок, підвищення читабельності програм;
* забезпечення зручного механізму передачі аргументів у шаблон класу з метою їх обробки методами класу.

**Додаток А**

Header.h

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Main {

public:

void main();

};

class Rectangle {

int a;

int b;

public:

Rectangle(int a = 0, int b = 0) { this->a = a; this->b = b; }

friend Rectangle& operator + (const Rectangle& ob1, const Rectangle& ob2) {

Rectangle result(ob1.a + ob2.a, ob1.b + ob2.b);

return result;

};

friend Rectangle& operator - (const Rectangle& ob1, const Rectangle& ob2) {

Rectangle result(ob1.a - ob2.a, ob1.b - ob2.b);

return result;

};

friend Rectangle& operator / (const Rectangle& ob1, const Rectangle& ob2) {

Rectangle result(ob1.a / ob2.a, ob1.b / ob2.b);

return result;

};

friend Rectangle& operator / (const Rectangle& ob1, int number) {

Rectangle result(ob1.a / number, ob1.b / number);

return result;

};

friend Rectangle& operator \* (const Rectangle& ob1, const Rectangle& ob2) {

Rectangle result(ob1.a \* ob2.a, ob1.b \* ob2.b);

return result;

};

bool operator == (const Rectangle& ob) { return ((a \* b) == (ob.a \* ob.b)); }

bool operator != (const Rectangle& ob) { return !((a \* b) == (ob.a \* ob.b)); }

bool operator > (const Rectangle& ob) { return ((a \* b) > (ob.a \* ob.b)); };

bool operator < (const Rectangle& ob) { return ((a \* b) < (ob.a \* ob.b)); };

bool operator >= (const Rectangle& ob) { return ((a \* b) >= (ob.a \* ob.b)); };

bool operator <= (const Rectangle& ob) { return ((a \* b) <= (ob.a \* ob.b)); };

Rectangle& operator = (const Rectangle& ob)

{

a = ob.a; b = ob.b;

return \*this;

}

void squrt()

{

this->a = sqrt(this->a);

this->b = sqrt(this->b);

}

friend ostream& operator <<(ostream& out, const Rectangle& ob) {

out << "A = " << ob.a << " || B = " << ob.b;

return out;

}

};

class Parallelepiped {

float a;

float b;

float c;

public:

Parallelepiped(float a = 0, float b = 0, float с = 0) { this->a = a; this->b = b; this->c = с;}

friend Parallelepiped& operator + (const Parallelepiped& ob1, const Parallelepiped& ob2) {

Parallelepiped result(ob1.a + ob2.a, ob1.b + ob2.b, ob1.c + ob2.c);

return result;

};

friend Parallelepiped& operator - (const Parallelepiped& ob1, const Parallelepiped& ob2) {

Parallelepiped result(ob1.a - ob2.a, ob1.b - ob2.b, ob1.c - ob2.c);

return result;

};

friend Parallelepiped& operator / (const Parallelepiped& ob1, const Parallelepiped& ob2) {

Parallelepiped result(ob1.a / ob2.a, ob1.b / ob2.b, ob1.c / ob2.c);

return result;

};

friend Parallelepiped& operator / (const Parallelepiped& ob1, int number) {

Parallelepiped result(ob1.a / number, ob1.b / number, ob1.c / number);

return result;

};

friend Parallelepiped& operator \* (const Parallelepiped& ob1, const Parallelepiped& ob2) {

Parallelepiped result(ob1.a \* ob2.a, ob1.b \* ob2.b, ob1.c \* ob2.c);

return result;

};

/\*Parallelepiped& operator = (const Parallelepiped& ob)

{

a = ob.a; b = ob.b; c = ob.c;

return \*this;

}\*/

bool operator == (const Parallelepiped& ob) { return ((a \* b \* c) == (ob.a \* ob.b \* ob.c)); }

bool operator != (const Parallelepiped& ob) { return !((a \* b \* c) == (ob.a \* ob.b \* ob.c)); }

bool operator > (const Parallelepiped& ob) { return ((a \* b \* c) > (ob.a \* ob.b \* ob.c)); };

bool operator < (const Parallelepiped& ob) { return ((a \* b \* c) < (ob.a \* ob.b \* ob.c)); };

bool operator >= (const Parallelepiped& ob) { return ((a \* b \* c) >= (ob.a \* ob.b \* ob.c)); };

bool operator <= (const Parallelepiped& ob) { return ((a \* b \* c) <= (ob.a \* ob.b \* ob.c)); };

void squrt()

{

this->a = sqrt(this->a);

this->b = sqrt(this->b);

this->c = sqrt(this->c);

}

friend ostream& operator <<(ostream& out, const Parallelepiped& ob) {

out << "a = " << ob.a << " || b = " << ob.b << " || c = " << ob.c << " || v = " << ob.a \* ob.b \* ob.c;

return out;

}

};

class Fraction {

float numerator;

float denominator;

public:

Fraction() { (this->denominator = rand() % 50 + 1) / (this->numerator = rand() % 50 + 1); };

Fraction(const float denominator, const float numerator) {

if (numerator != 0)

(this->denominator = denominator) / (this->numerator = numerator);

else

cout << "\tYou whant to brake the math rules? -\_-\n";

};

// stream operators

friend ostream& operator << (ostream& out, const Fraction& ob) { out << ob.denominator << " / " << ob.numerator << " = " << setprecision(10) << ob.denominator / ob.numerator; return out; };

//friend istream& operator >> (istream& in, Fraction& ob) { cout << "Denominator --> "; in >> ob.denominator; cout << "Numerator --> "; in >> ob.numerator; return in; };

// ar operation

friend Fraction& operator+(const Fraction& ob1, const Fraction& ob2)

{

Fraction result((ob1.denominator \* ob2.numerator) + (ob2.denominator \* ob1.numerator), ob1.numerator \* ob2.numerator);

return result;

}

friend Fraction& operator-(const Fraction& ob1, const Fraction& ob2)

{

Fraction result((ob1.denominator \* ob2.numerator) - (ob2.denominator \* ob1.numerator), ob1.numerator \* ob2.numerator);

return result;

}

friend Fraction& operator/(const Fraction& ob1, const Fraction& ob2)

{

Fraction result(ob2.numerator \* ob1.denominator, ob2.denominator \* ob1.numerator);

return result;

}

friend Fraction& operator / (Fraction& ob1, int number)

{

Fraction result(ob1.denominator, ob1.numerator\*3);

return result;

}

friend Fraction& operator\*(const Fraction& ob1, const Fraction& ob2)

{

Fraction result(ob1.denominator \* ob2.denominator, ob1.numerator \* ob2.numerator);

return result;

}

Fraction& operator = (const Fraction& ob)

{

denominator = ob.denominator; numerator = ob.numerator;

return \*this;

}

void squrt()

{

this->numerator = sqrt(this->numerator);

this->denominator = sqrt(this->denominator);

}

//compering operations

bool operator == (const Fraction& ob) { return (denominator / numerator == ob.denominator / ob.numerator); }

bool operator != (const Fraction& ob) { return !(denominator / numerator == ob.denominator / ob.numerator); }

bool operator > (const Fraction& ob) { return (denominator / numerator > ob.denominator / ob.numerator); };

bool operator < (const Fraction& ob) { return (denominator / numerator < ob.denominator / ob.numerator); };

bool operator >= (const Fraction& ob) { return (denominator / numerator >= ob.denominator / ob.numerator); };

bool operator <= (const Fraction& ob) { return (denominator / numerator <= ob.denominator / ob.numerator); };

};

template <class T>

class Task1

{

T ob1, ob2, ob3;

public:

Task1(){}

Task1(T a, T b, T c)

{

this->ob1 = a;

this->ob2 = b;

this->ob3 = c;

}

void maxim()

{

T arry[3] = { ob1, ob2, ob3 };

T max = arry[0];

for (int j = 0; j < 3; j++)

if (arry[j] > arry[j - 1])

max = arry[j];

cout << "The maximal is: " << max << endl;

}

void minim()

{

T arry[3] = { ob1, ob2, ob3 };

T min = arry[0];

for (int j = 0; j < 3; j++)

if (arry[j] < arry[j - 1])

min = arry[j];

cout << "The minimal is: " << min << endl;

}

void arith\_mean()

{

T temp = ob1 + ob2;

temp = temp + ob3;

temp = temp / 3;

cout << "Arithmetical mean: " << temp << endl;

}

void sqrt\_mean()

{

T t1 = ob1 \* ob1;

T t2 = ob2 \* ob2;

T t3 = ob3 \* ob3;

T t4 = t1 + t2;

T t5 = t4 + t3;

T temp = t5 / 3;

temp.squrt();

cout << "Medium-square mean: " << temp << endl;

}

void summary()

{

T t1 = ob1 + ob2;

t1 = t1 + ob3;

cout << "The summary is: " << t1 << endl;

}

void multipliance()

{

T t1 = ob1 \* ob2;

t1 = t1 \* ob3;

cout << "The multipliance is: " << t1 << endl;

}

};

template <class T>

class Task2

{

int size;

T\* ob;

public:

Task2()

{

this->ob = nullptr;

this->size = 0;

}

Task2(T\* obj, int size)

{

this->size = size;

this->ob = obj;

}

void maxim()

{

T max = ob[0];

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

{

if (max <= ob[i])

max = ob[i];

}

cout << "The maximal element:\n";

cout << max << endl;

}

void minim()

{

T min = ob[0];

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

{

if (min >= ob[i])

min = ob[i];

}

cout << "The minimal element:\n";

cout << min << endl;

}

void sum()

{

T sum = ob[0];

for (int i = 1; i < size; i++)

sum = sum + ob[i];

cout << "The summary is:\n";

cout << sum << endl;

}

void mult()

{

T mult = ob[0];

for (int i = 1; i < size; i++)

mult = mult \* ob[i];

cout << "The multipliance is:\n";

cout << mult << endl;

}

void mean()

{

T sum = ob[0];

for (int i = 1; i < size; i++)

sum = sum + ob[i];

sum = sum / this->size;

cout << "The mean is:\n";

cout << sum << endl;

}

void sq\_mean()

{

T sum = ob[0] \* ob[0];

for (int i = 1; i < size; i++){

T mult = ob[i] \* ob[i];

sum = sum + mult;

}

sum = sum / this->size;

sum.squrt();

cout << "The square mean is:\n";

cout << sum << endl;

}

void filter(T criteria)

{

int cnt = 0;

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

if (criteria == ob[i]) {

cnt++;

cout << i + 1 << ") " << ob[i] << endl;

}

if (cnt != 0)

cout << "There are " << cnt << " coincidenses\n";

else

cout << "Coincidenses hasn`t found!\n";

}

};

**Додаток Б**

Source.cpp

#include "Header.h"

void Main::main()

{

srand(time(NULL));

int button = -1;

while (true) { // while stage 1 menu

cout << "Choose task: \n"

<< "By 3 objects - 1\n"

<< "By array of objects - 2\n"

<< "Exit - 0\n"

<< "--> "; cin >> button;

switch (button) // first stage of menu

{

////////////////// 1 case 1 stage ////////////////////////////

case 1:

system("cls");

while (button != 0)

{

cout << "Choose the class for work: \n"

<< "Rectangle - 1 \n"

<< "Parallelepiped - 2 \n"

<< "Fraction - 3 \n"

<< "Back - 0 \n"

<< "--> "; cin >> button;

if (button == 1) {

system("cls");

Rectangle ob1(rand() % 50 + 1, rand() % 50 + 1), ob2(rand() % 50 + 1, rand() % 50 + 1), ob3(rand() % 50 + 1, rand() % 50 + 1);

Task1 <Rectangle> T1rec(ob1, ob2, ob3);

cout << "\nRectangle: \n";

cout << ob1 << endl << ob2 << endl << ob3 << endl << endl;

T1rec.maxim();

T1rec.minim();

T1rec.arith\_mean();

T1rec.multipliance();

T1rec.sqrt\_mean();

T1rec.summary();

system("pause");

}

else if (button == 2) {

system("cls");

Parallelepiped ob1(rand() % 10 + 1, rand() % 10 + 1, rand() % 10 + 1), ob2(rand() % 10 + 1, rand() % 10 + 1, rand() % 10 + 1), ob3(rand() % 10 + 1, rand() % 10 + 1, rand() % 10 + 1);

Task1 <Parallelepiped> T1par(ob1, ob2, ob3);

cout << "\Parallelepiped: \n";

cout << ob1 << endl << ob2 << endl << ob3 << endl << endl;

T1par.maxim();

T1par.minim();

T1par.arith\_mean();

T1par.multipliance();

T1par.sqrt\_mean();

T1par.summary();

system("pause");

}

else if (button == 3) {

system("cls");

Fraction ob1(rand() % 50 + 1, rand() % 50 + 1), ob2(rand() % 50 + 1, rand() % 50 + 1), ob3(rand() % 50 + 1, rand() % 50 + 1);

Task1 <Fraction> T1frac(ob1, ob2, ob3);

cout << "\Fraction: \n";

cout << ob1 << endl << ob2 << endl << ob3 << endl << endl;

T1frac.maxim();

T1frac.minim();

T1frac.arith\_mean();

T1frac.multipliance();

T1frac.sqrt\_mean();

T1frac.summary();

system("pause");

}

system("cls");

}

break;

////////////////// 2 case 1 stage ////////////////////////////

case 2:

int size;

system("cls");

while (button != 0)

{

cout << "Choose the class for work: \n"

<< "Rectangle - 1 \n"

<< "Parallelepiped - 2 \n"

<< "Fraction - 3 \n"

<< "Back - 0 \n"

<< "--> "; cin >> button;

if (button == 1) {

system("cls");

cout << "Choose the size of array --> "; cin >> size;

Rectangle\* recarray = new Rectangle[size];

for (size\_t i = 0; i < size; i++)

recarray[i] = { rand() % 50 + 1, rand() % 50 + 1 };

cout << "Rectangle: \n";

for (size\_t i = 0; i < size; i++)

cout << i + 1 << ") " << recarray[i] << endl;

Task2 <Rectangle> T2rec(recarray, size);

T2rec.maxim();

T2rec.minim();

T2rec.sum();

T2rec.mult();

T2rec.mean();

T2rec.sq\_mean();

int a, b;

cout << "Find equal elements, enter side A first then side B -->"; cin >> a >> b;

Rectangle key(a, b);

T2rec.filter(key);

system("pause");

delete[] recarray; recarray = NULL;

}

else if (button == 2) {

system("cls");

cout << "Choose the size of array --> "; cin >> size;

Parallelepiped\* paraarray = new Parallelepiped[size];

for (size\_t i = 0; i < size; i++)

paraarray[i] = { (float)(rand() % 10 + 1), (float)(rand() % 10 + 1), (float)(rand() % 10 + 1) };

cout << "Parallelepiped: \n";

for (size\_t i = 0; i < size; i++)

cout << i + 1 << ") " << paraarray[i] << endl;

Task2 <Parallelepiped> T2para(paraarray, size);

T2para.maxim();

T2para.minim();

T2para.sum();

T2para.mult();

T2para.mean();

T2para.sq\_mean();

int a, b, c;

cout << "Find equal elements, enter side A first then side B and then side C -->"; cin >> a >> b >> c;

Parallelepiped key(a, b, c);

T2para.filter(key);

system("pause");

delete[] paraarray; paraarray = NULL;

}

else if (button == 3) {

system("cls");

cout << "Choose the size of array --> "; cin >> size;

Fraction\* fracarray = new Fraction[size];

for (size\_t i = 0; i < size; i++)

fracarray[i] = { (float)(rand() % 50 + 1), (float)(rand() % 50 + 1) };

cout << "Fraction: \n";

for (size\_t i = 0; i < size; i++)

cout << i + 1 << ") " << fracarray[i] << endl;

Task2 <Fraction> T2frac(fracarray, size);

T2frac.maxim();

T2frac.minim();

T2frac.sum();

T2frac.mult();

T2frac.mean();

T2frac.sq\_mean();

int a, b;

cout << "Find equal elements, enter side numerator first then side denominator -->"; cin >> a >> b;

Fraction key(a, b);

T2frac.filter(key);

system("pause");

delete[] fracarray; fracarray = NULL;

}

system("cls");

}

break;

////////////////// 0 case 1 stage ////////////////////////////

case 0:

system("cls");

exit(0);

break;

default:

system("cls");

break;

}

}

}