Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

до лабораторної роботи № 5

з дисципліни ООП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *ІП-61 Кушка Михайло Олександрович* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *Головченко М.М.* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

ЗМІСТ

1. Мета роботи 3

2. Постановка задачі 4

3. Аналітичні викладки 5

4. UML-діаграма класів 6

5. Вихідний код програми 7

prototypes.cpp 7

prototypes.hpp 8

functions.cpp 9

functions.hpp 10

stdafx.hpp 10

main.cpp 11

6. Приклади роботи програми 12

7. Висновки 13

# Мета роботи

Мета роботи - вивчити особливості переривань у мові програмування С++. Навчитися “відловлювати” помилки, що можуть виникнути під час роботи програми і створювати для них власні обробники переривань для уникнення аварійного завершення програми.

# Постановка задачі

Спроектувати клас «Vector», який містить координати вектора в просторі. Для нього визначити: операцію складання «+», операцію віднімання «-», операцію векторного множення «\*», операцію множення на константу «\*», ці ж операції в скороченій формі, операцію унарний мінус «-», логічну операцію «==», яка перевіряє вектори на колінеарність. При необхідності дозволяється визначати інші операції (наприклад «=») і методи (наприклад, getter, setter та інше). Продемонструвати кожну операцію. Розглянути випадок коли вектор заданий на площині.

# Аналітичні викладки

Виняткова ситуація - це незвичайні або несподівані обставини, що виникають в роботі програми: збої апаратури, помилки введення-виведення, програмні помилки. Виняткові ситуації призводять до переривання програми, після чого слід виявити виключення і обробити його таким чином, щоб виключити переривання при виконанні програми.

Для роботи з виключеннями в С ++ використовуються такі оператори:

 try - початок блоку винятків;

 catch - початок блоку, який займається обробкою виключення.

Throw – збудження (створення) виключення. Даний оператор може використовуватися в декількох форматах:

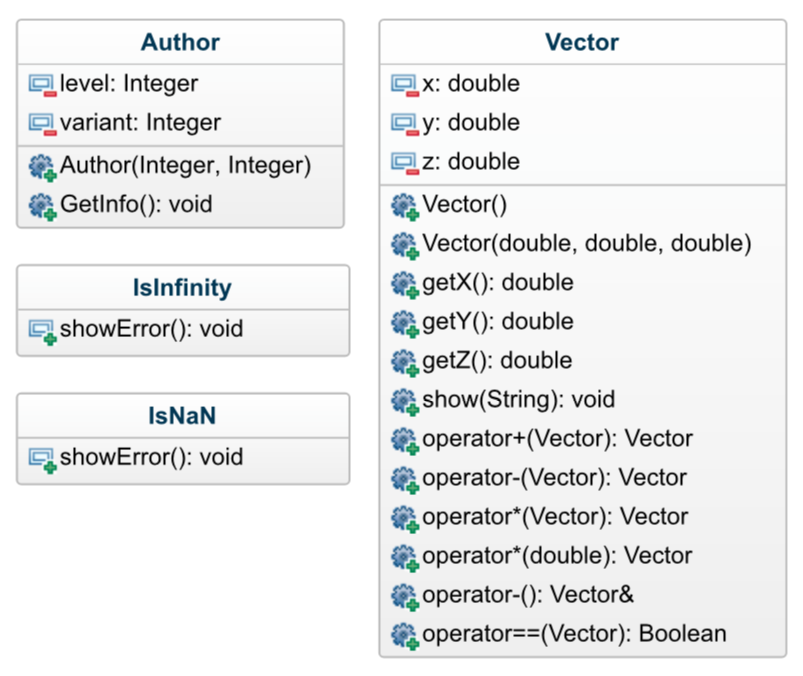
 throw (type) - виключення заданого типу;

 throw (type1, type2, ...) - створення виключення, залежить від декількох типів

 throw клас (аргументи) - виключення заданого класу

 throw () - не генерується виключення.

# UML-діаграма класів



# Вихідний код програми

## prototypes.cpp

//

// prototypes.cpp

// Lab4

//

// Created by Kushka Misha on 10/28/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#include "prototypes.hpp"

#include "errors.hpp"

// Sum

Vector Vector::operator+(Vector vec) {

Vector temp;

temp.x = x + vec.x;

temp.y = y + vec.y;

temp.z = z + vec.z;

return temp;

}

// Substraction

Vector Vector::operator-(Vector vec) {

Vector temp;

temp.x = x - vec.x;

temp.y = y - vec.y;

temp.z = z - vec.z;

return temp;

}

// Multiplication of a 2 vectors

Vector Vector::operator\*(Vector vec) {

Vector temp;

temp.x = y \* vec.z - z \* vec.y;

temp.y = z \* vec.x - x \* vec.z;

temp.z = x \* vec.y - y \* vec.x;

return temp;

}

// Multiplication by a constant

Vector Vector::operator\*(float value) {

Vector temp;

temp.x = x \* value;

temp.y = y \* value;

temp.z = z \* value;

return temp;

}

// Unary minus

Vector& Vector::operator-() {

Vector temp;

temp.x = -x;

temp.y = -y;

temp.z = -z;

return temp;

}

// Collinearity

bool Vector::operator==(Vector vec) {

if (x == 0 && vec.x == 0) {

if (y / vec.y == z / vec.z)

return true;

return false;

} else if (y == 0 && vec.y == 0) {

if (x / vec.x == z / vec.z)

return true;

return false;

} else if (z == 0 && vec.z == 0) {

if (x / vec.x == y / vec.y)

return true;

return false;

} else {

float alpha = 0;

alpha = x / vec.x;

if (y / vec.y == alpha && z / vec.z == alpha)

return true;

}

return false;

}

// Show vector

void Vector::show(string s) {

try {

if (isinf(x) || isinf(y) || isinf(z)) {

throw IsInfinity();

}

if (isnan(x) || isnan(y) || isnan(z)) {

throw IsNaN();

}

cout << s << " = (" << x <<", " << y << ", " << z << ")" << endl;

} catch (IsInfinity &e) {

e.showError();

cout << s << " = (" << x <<", " << y << ", " << z << ")" << endl;

} catch (IsNaN &e) {

e.showError();

}

}

void Author::GetInfo() {

// Displays author info.

cout << "\

-----------------------\n\

| Kushka Misha, IP-61 |\n\

| Level: " << level << " |\n\

| Variant: " << variant << " |\n\

-----------------------\n\n";

}

## prototypes.hpp

//

// prototypes.hpp

// Lab4

//

// Created by Kushka Misha on 10/28/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#ifndef prototypes\_hpp

#define prototypes\_hpp

#include "stdafx.hpp"

class Vector {

double x, y, z;

public:

Vector(double a, double b, double c) : x(a), y(b), z(c) {}; // Constructor

Vector() { x = y = z = 0; } // Blank constructor

// ~Vector() { cout << "Destructor..." << endl; } // Destructor

double getX() { return x; }

double getY() { return y; }

double getZ() { return z; }

void show(string s) {

cout << s << " = (" << x <<", " << y << ", " << z << ")" << endl;

}

Vector operator+(Vector); // Sum of 2 vectors

Vector operator-(Vector); // Substraction of a 2 vectors

Vector operator\*(Vector); // Multiplication of a 2 vectors

Vector operator\*(double); // Multiplication by a constant

Vector& operator-(); // Unary minus operation

bool operator==(Vector); // Check vectors collinearity

};

// Class to display some useful info about author of the program.

class Author {

int level, variant;

public:

Author(int level=3, int variant=15) : level(level), variant(variant) {}

void GetInfo();

};

#endif /\* prototypes\_hpp \*/

## functions.cpp

//

// functions.cpp

// Lab4

//

// Created by Kushka Misha on 10/28/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#include "functions.hpp"

// Show all class operator functions

void show\_demo(Vector a, Vector b, Vector c) {

// c = a + b

cout << "c = a + b" << endl;

c = a + b;

c.show("C");

cout << endl;

// c = a - b

cout << "c = a - b" << endl;

c = a - b;

c.show("C");

cout << endl;

// c = a \* b

cout << "c = a \* b" << endl;

c = a \* b;

c.show("C");

cout << endl;

// c = a \* const

cout << "c = a \* 3" << endl;

c = a \* 3;

c.show("C");

cout << endl;

// c = -a

cout << "c = -a" << endl;

c = -a;

c.show("C");

cout << endl;

// isCollinear = a == b

bool isCollinear = false;

cout << "isCollinear = a == b" << endl;

isCollinear = a == b;

cout << "isCollinear == " << isCollinear << endl << endl;

// isCollinear = a == a\*2

cout << "isCollinear = a == a\*2" << endl;

isCollinear = a == a\*2;

cout << "isCollinear == " << isCollinear << endl << endl;

}

// Check input values for vector

values input() {

string xs, ys, zs;

values val;

while (true) {

// Enter x

while(true) {

try {

cout << "Enter x\n> ";

cin >> xs;

val.x = stod(xs);

break;

} catch(...) {

cout << "Error! Please enter number, not string." << endl;;

}

}

// Enter y

while(true) {

try {

cout << "Enter y\n> ";

cin >> ys;

val.y = stod(ys);

break;

} catch(...) {

cout << "Error! Please enter number, not string." << endl;;

}

}

// Enter z

while(true) {

try {

cout << "Enter z\n> ";

cin >> zs;

val.z = stod(zs);

break;

} catch(...) {

cout << "Error! Please enter number, not string." << endl;;

}

}

// Is zero vector

if (val.x == 0 && val.y == 0 && val.z == 0) {

cout << "Length of vector can't be zero (0, 0, 0). Please try another values." << endl << endl;

} else {

break;

}

}

return val;

}

## functions.hpp

//

// functions.hpp

// Lab4

//

// Created by Kushka Misha on 10/28/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#ifndef functions\_hpp

#define functions\_hpp

#include "prototypes.hpp"

void show\_demo(Vector, Vector, Vector);

values input();

#endif /\* functions\_hpp \*/

## errors.hpp

//

// errors.hpp

// Lab5

//

// Created by Kushka Misha on 12/4/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#ifndef errors\_hpp

#define errors\_hpp

class IsInfinity {

public:

void showError() {

cout << "Used type is too small to display so big value as following, so it'll be displayed as 'inf'." << endl;

}

};

class IsNaN {

public:

void showError() {

cout << "Too big value" << endl;

}

};

#endif /\* errors\_hpp \*/

## stdafx.hpp

//

// stdafx.hpp

// Lab4

//

// Created by Kushka Misha on 10/28/17.

// Copyright © 2017 Kushka Misha. All rights reserved.

//

#ifndef stdafx\_hpp

#define stdafx\_hpp

#include <iostream>

#include <string>

using namespace std;

#endif /\* stdafx\_hpp \*/

## main.cpp

#include "prototypes.hpp"

#include "functions.hpp"

int main() {

// Author information

Author \*auth = new Author();

auth->GetInfo();

// Input

string xs, ys, zs;

values valA, valB;

string cont = "y";

while(cont == "y") {

cout << "=== First vector ===" << endl << endl;

valA = input();

cout << "=== Second vector ===" << endl << endl;

valB =input();

Vector a(valA.x, valA.y, valA.z),

b(valB.x, valB.y, valB.z),

c;

// Demo

cout << endl;

a.show("A");

b.show("B");

cout << endl;

show\_demo(a, b, c);

cout << endl << endl << "Continue? (y / n)\n>";

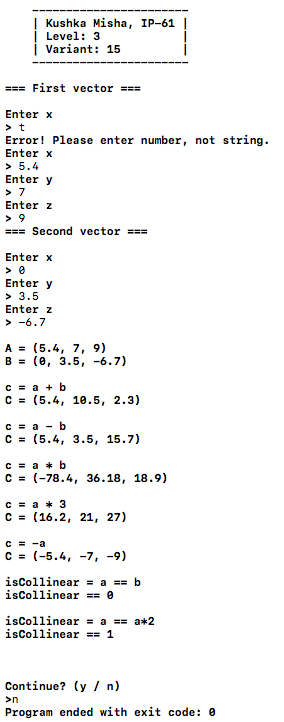
cin >> cont;

}

return 0;

}

# Приклади роботи програми



# Висновки

У даній лабораторній роботі я використав обробку виключень для запобігання виникнення помилок у роботі програми при введенні некоректних даних. Також це дозволило запобігти аварійному “вильоту” програми.