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 № 2

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

6. Приклади роботи програми 12

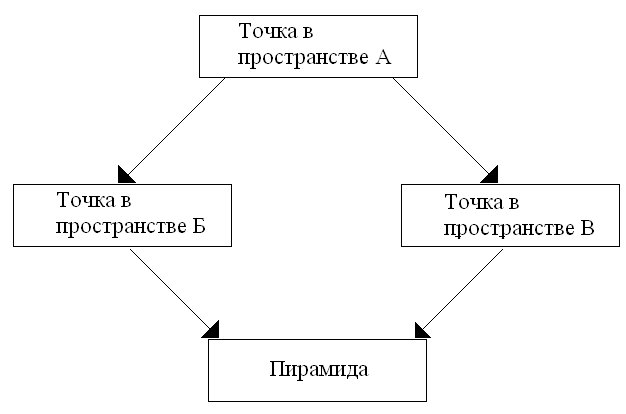
7. Висновки 13

# Мета роботи

Мета роботи - вивчити основні концепції об'єктно-орієнтованого програмування. Вивчити особливості успадкування і множинного спадкоємства.

# Постановка задачі

Визначити наступну ієрархію класів:



Визначити довжину ребер і рівняння площини основи піраміди. Ініціалізація даних проводиться користувачем.

# Аналітичні викладки

Наслідування - це здатність брати існуючий (базовий) клас і породжувати з нього новий клас - нащадок, з успадкуванням всіх його атрибутів і поведінки.

Синтаксис визначення похідного класу:

class ім'я\_класу: спеціфікатор\_доступу1 базовий\_клас1, спеціфікатор\_доступу2 базовий\_клас2, {

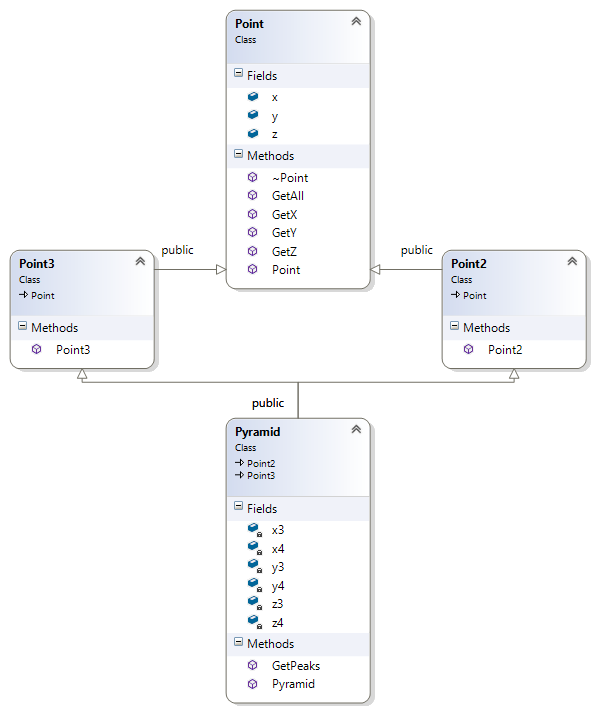
   // тіло класу

};

При описі нового класу, похідного від якогось одного або декількох базових класів, можна додавати нові функції-елементи і елементи-дані, зберігаючи при цьому всі елементи батьків, а можна батьківські елементи перевизначити або перевантажити. У похідному класі доступні відкриті і захищені елементи базового класу (прямого або непрямого). Закриті елементи базового класу в похідному класі недоступні.

Похідний клас може успадковуватися від базового класу як public, protected або private. Захищене і закрите спадкоємство зустрічаються рідко і кожне з них потрібно використовувати з великою обережністю.

# UML-діаграма класів



# Вихідний код програми

## prototypes.cpp

//

//  prototypes.cpp

//  Lab2

//

//  Created by Kushka Misha on 9/16/17.

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

//

#include "prototypes.hpp"

void Point::GetX() {

    cout << "x = " << x << endl;

}

void Point::GetY() {

    cout << "y = " << y << endl;

}

void Point::GetZ() {

    cout << "z = " << z << endl;

}

void Point::GetAll() {

    cout << "x = " << x << "; y = " << y << "; z = " << z << endl;

}

Point::~Point() {

    // Point class destructor.

    cout << "Delete something..." << endl;

}

void Pyramid::GetPeaks() {

    cout << endl << "--x--|" << "--y--|" << "--z--" << endl;

    cout << setw(4) << x1 << " |" << setw(4) << y1 << " |" << setw(4) << z1 << endl;

    cout << setw(4) << x2 << " |" << setw(4) << y2 << " |" << setw(4) << z2 << endl;

    cout << setw(4) << x3 << " |" << setw(4) << y3 << " |" << setw(4) << z3 << endl;

    cout << setw(4) << x4 << " |" << setw(4) << y4 << " |" << setw(4) << z4 << endl;

}

void Pyramid::GetEdges() {

    // Calculate edges length of the base of the pyramid.

    cout << "\nLength of the first edge = " << sqrt(pow(x3 - x2, 2) + pow(y3 - y2, 2) + pow(z3 - z2, 2)) << "." << endl;

    cout << "Length of the second edge = " << sqrt(pow(x4 - x3, 2) + pow(y4 - y3, 2) + pow(z4 - z3, 2)) << "." << endl;

    cout << "Length of the third edge = " << sqrt(pow(x4 - x2, 2) + pow(y4 - y2, 2) + pow(z4 - z2, 2)) << ".\n" << endl;

}

void Pyramid::PlaneEquation() {

    // Calculate plane equation for the base of the pyramid.

    double A = y2\*(z3 - z4) + y3\*(z4 - z2) + y4\*(z2 - z3);

    double B = z2\*(x3 - x4) + z3\*(x4 - x2) + z4\*(x2 - x3);

    double C = x2\*(y3 - y4) + x3\*(y4 - y2) + x4\*(y2 - y3);

    double D = -A \* x2 -B \* y2 - C \* z2;

    string a = " + ";

    string b = " + ";

    string c = " + ";

    if (B < 0) {

        a = " - ";

        B = -B;

    }

    if (C < 0) {

        b = " - ";

        C = -C;

    }

    if (D < 0) {

        c = " - ";

        D = -D;

    }

    cout << "The plane equation is: " << A << "x" << a << B << "y" << b << C << "z" << c << D << " = 0." << endl;

}

void Author::GetInfo() {

    // Displays author info.

    cout << "\

    -----------------------\n\

    | Kushka Misha, IP-61 |\n\

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

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

    -----------------------\n\n";

}

int\* input\_coordinates() {

    // Get coordinates of the vertexes via user input.

    int \*arr = new int[12];

    cout << "Enter x, y, z of the first point, separated by spaces\n> ";

    cin >> arr[0] >> arr[1] >> arr[2];

    cout << "Enter x, y, z of the second point, separated by spaces\n> ";

    cin >> arr[3] >> arr[4] >> arr[5];

    cout << "Enter x, y, z of the third point, separated by spaces\n> ";

    cin >> arr[6] >> arr[7] >> arr[8];

    cout << "Enter x, y, z of the last point, separated by spaces\n> ";

    cin >> arr[9] >> arr[10] >> arr[11];

    return arr;

}

int get\_choice() {

    // Simple menu for methods of the Pyramid class.

    int ch = 0;

    while (ch != 1 && ch != 2 && ch != 3 && ch != 4) {

        cout << "\nWhat do you want?\n";

        cout << "\t1 - Show all point coordinates\n";

        cout << "\t2 - Show length of edges in the base of the pyramid\n";

        cout << "\t3 - Show equation of the plane in the base of the pyramid\n";

        cout << "\t4 - Do all of the above\n";

        cin >> ch;

        if (ch != 1 && ch != 2 && ch != 3 && ch != 4)

            cout << "Invalid choice, try again." << endl;

    }

    return ch;

}

## prototypes.hpp

//

//  prototypes.hpp

//  Lab2

//

//  Created by Kushka Misha on 9/16/17.

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

//

#ifndef prototypes\_hpp

#define prototypes\_hpp

#include "stdafx.hpp"

// Emulation of the point.

class Point {

public:

    int x, y, z;

    Point(int x, int y, int z) : x(x), y(y), z(z) {}

    void GetX();

    void GetY();

    void GetZ();

    void GetAll();

    ~Point();

};

class Point2: public Point {

public:

    Point2(int x, int y, int z) : Point(x, y, z) {}

};

class Point3: public Point {

public:

    Point3(int x, int y, int z) : Point(x, y, z) {}

};

// Emulation of the pyramid.

class Pyramid: public Point2, public Point3  {

    int x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4;

public:

    Pyramid(int x10, int y10, int z10, int x20, int y20, int z20, int x3, int y3, int z3, int x4, int y4, int z4) :

    Point2(x10, y10, z10), Point3(x20, y20, z20), x3(x3), y3(y3), z3(z3), x4(x4), y4(y4), z4(z4) {

        x1 = Point2::x;

        y1 = Point2::y;

        z1 = Point2::z;

        x2 = Point3::x;

        y2 = Point3::y;

        z2 = Point3::z;

    }

    void GetPeaks();

    void GetEdges();

    void PlaneEquation();

};

// 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();

};

// Get coordinates of the vertexes via user input.

int\* input\_coordinates();

// Simple menu for methods of the Pyramid class.

int get\_choice();

#endif /\* prototypes\_hpp \*/

## stdafx.hpp

//

//  stdafx.hpp

//  Lab2

//

//  Created by Kushka Misha on 9/20/17.

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

//

#ifndef stdafx\_hpp

#define stdafx\_hpp

#include <iostream>

#include <iomanip>

#include <time.h>

#include <string>

#include <math.h>

using namespace std;

#endif /\* stdafx\_hpp \*/

## main.cpp

#include "prototypes.hpp"

int main() {

    srand(time(NULL));

    // Display some usefull info.

    Author \*auth = new Author;

    auth->GetInfo();

    Point \*p = new Point(5, 10, 3);

    p->GetAll();

    delete p;

    Point2 \*p2 = new Point2(-2, 2, 0);

    p2->GetAll();

    delete p2;

    Point3 \*p3 = new Point3(0, 11, 9);

    p3->GetAll();

    delete p3;

    Pyramid \*pir;

    char cont\_program;

    char cont\_this;

    int \*arr;

    while (true) {

//        arr = input\_coordinates();

//        pir = new Pyramid(arr[0], arr[1], arr[2], arr[3], arr[4], arr[5], arr[6], arr[7], arr[8], arr[9], arr[10], arr[11]);

        pir = new Pyramid(1, 2, 1, 5, 6, -7, 8, 0, 0, 3, 4, 2);

        while (true) {

            int choice = get\_choice();

            if (choice == 1)

                pir->GetPeaks();

            else if (choice == 2)

                pir->GetEdges();

            else if (choice == 3)

                pir->PlaneEquation();

            else if (choice == 4) {

                pir->GetPeaks();

                pir->GetEdges();

                pir->PlaneEquation();

            }

            cout << "Continue with this points? (y/n)\n> ";

            cin >> cont\_this;

            if (cont\_this == 'n')

                break;

        }

        cout << "Try enter another points? (y/n)\n> ";

        cin >> cont\_program;

        if (cont\_program == 'n')

            break;

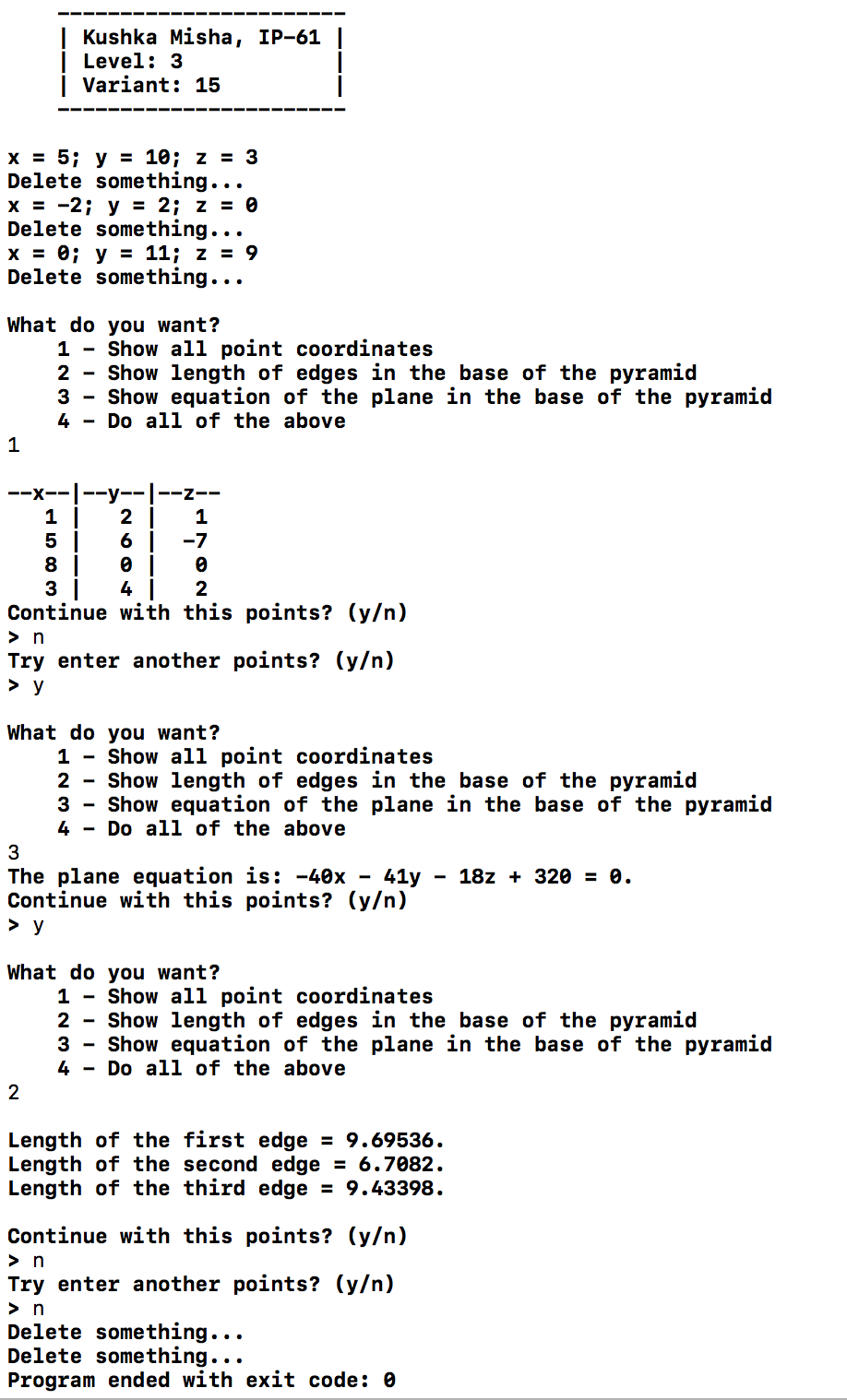
    }

    delete pir;

    return 0;

}

# Приклади роботи програми



# Висновки

У даній лабораторній роботі я познайомився з такою властивістю класів як наслідування та навчився його використовувати на мові програмування C++ у реальних задачах.