Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

**Лабораторна робота №1**

**Тема:** «Основні поняття ООП. Класи та об’єкти.

Функції доступу. Вказівник this»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Горбенко К.О. |
| Перевірив: Козірова Н.Л. |
|  |

Кропивницький

2024

**Лабораторна робота №1**

**Тема:** Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this

**Мета:**ознайомитись з основними поняттями ООП. Вивчити поняття

клас, об’єкт, сеттер, геттер та навчитись їх програмно реалізовувати мовою

С++.

**Варіант:** **3**

***Завдання 1:***

*1. Створіть клас "Person" (людина), який має наступні властивості та*

*функціональність:*

*Приватні поля класу:*

*• name (ім'я людини)*

*• age (вік людини)*

*• address (адреса людини)*

*Публічні методи класу:*

*• Метод setName(), який дозволяє задати ім'я людини.*

*• Метод getName(), який повертає ім'я людини.*

*• Метод setAge(), який дозволяє задати вік людини.*

*• Метод getAge(), який повертає вік людини.*

*• Метод setAddress(), який дозволяє задати адресу людини.*

*• Метод getAddress(), який повертає адресу людини.*

*2. Створіть об'єкт класу "Person".*

*3. Задайте значення полів об'єкта за допомогою відповідних методів.*

*4. Виведіть інформацію про людину на екран, використовуючи методи*

*для отримання значень полів.*

*5. Реалізувати програму за допомогою роздільної компіляції.*

*У вашому рішенні можуть бути додаткові методи та поля, якщо ви*

*вважаєте їх необхідними.*

***Лістинг Task\_1.cpp:***

#include "Person.h"

#include <iostream>

using namespace std;

int main() {

Person person;

person.setName("Alex");

person.setAge(25);

person.setAddress("Uman, st. Solnechnaya");

cout << "Name: " << person.getName() << endl;

cout << "Age: " << person.getAge() << endl;

cout << "Address: " << person.getAddress() << endl;

return 0;

}

***Лістинг Person.cpp:***

#include "Person.h"

using namespace std;

void Person::setName(const string& newName)

{

name = newName;

}

void Person::setAge(int newAge)

{

age = newAge;

}

void Person::setAddress(const string& newAddress)

{

address = newAddress;

}

string Person::getName() const

{

return name;

}

int Person::getAge() const

{

return age;

}

string Person::getAddress() const

{

return address;

}

***Лістинг Person.h:***

#ifndef PERSON\_H

#define PERSON\_H

#include <string>

using namespace std;

class Person {

private:

string name;

int age;

string address;

public:

void setName(const string& newName);

void setAge(int newAge);

void setAddress(const string& newAddress);

string getName() const;

int getAge() const;

string getAddress() const;

};

#endif

***Результат:***

**

***Завдання 2:***

*Реалізувати вище наведену задачу за допомогою структурного*

*програмування. У висновку описати різницю цих методів.*

***Лістинг Task\_2.cpp:***

#include <iostream>

#include <string>

using namespace std;

int main()

{

struct Person

{

string name;

int age;

string address;

};

Person firstPerson;

firstPerson.name = "George";

firstPerson.age = 21;

firstPerson.address = "Wroclaw, Jesionowa 21";

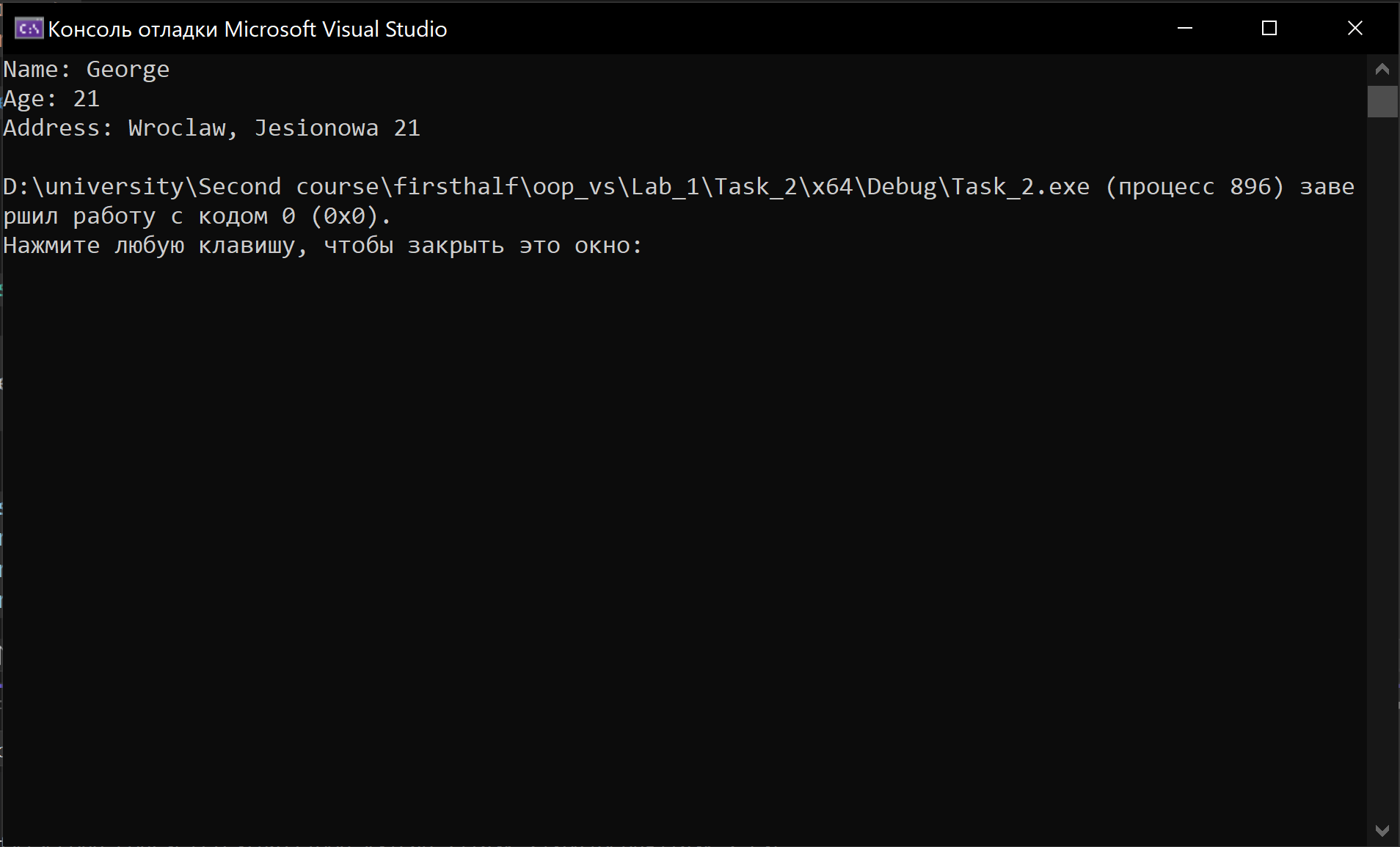
cout << "Name: " << firstPerson.name << endl;

cout << "Age: " << firstPerson.age << endl;

cout << "Address: " << firstPerson.address << endl;

}

***Результат:***

******

***Висновок:***

Виконавши лабораторну роботу, я ознайомилася з основними поняттями об'єктно-орієнтованого програмування, зокрема з поняттями класу, об'єкта, сеттера і геттера. У процесі роботи я створила клас "Person" з приватними полями для імені, віку та адреси, і реалізувала публічні методи для роботи з цими полями. Виконання завдань дозволило мені поглибити знання щодо структури класів і організації програмного коду. Підхід ООП виявився кращим за структурне програмування тим, що забезпечує кращу модульність і зручність масштабування. Використовуючи класи, можна легко об'єднувати дані та функціонал в одному обєкті, що робить код більш гнучким і легким для розуміння.