Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра ОТ

ЗВІТ

про виконання комп’ютерного практикуму № 4

з дисципліни

“Основи об'єктно-орієнтованого програмування”

Тема: ПОЛІМОРФІЗМ В С++

|  |  |  |
| --- | --- | --- |
| Прийняв: Головченко М.М. |  | Виконав: |
|  |  | студент 2-го курсу  гр. ІП-44 ФІОТ  Майоров І.М.  Варіант 8 |

Київ 2015

1. **ЗАДАНИЕ**

8. Спроектировать иерархию классов: класс фигуры в пространстве и его наследники: тетраэдр, куб, параллелепипед. Определить в базовом классе и переопределить в наследниках методы вычисления полной площади поверхности и объема фигуры. Элементы-данные фигур объявляются в базовом классе, а инициализируются в наследниках (элементы данные: стороны)

1. **ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ**

Цель работы – изучить основные концепции объектно-ориентированного программирования. Изучить особенности виртуальных функций, абстрактных классов и полиморфизма.

1. **АНАЛИТИЧЕСКИЕ ИЗЛАГАНИЯ**

Полиморфизм – свойство кода С++ вести себя по-разному в зависимости от ситуации, возникающей в момент выполнения; это возможность объектов разных классов, связанных наследованием, реагировать различным образом при обращении к одной и той же функции-элементу с одинаковым именем и разным определением. Возникает проблема распознавания функций, решаемая при помощи раннего (статического) связывания или позднего (динамического). Раннее связывание происходит на этапе компиляции, путём вызова фиксированных идентификаторов функций компилятором, замены идентификаторов данными (физическими адресами) компоновщиком задач.

Позднее связывание – при выполнении программы, выбор нужной функции – самой программой. Возможно для объектов связанных иерархией и реализуется через механизм виртуальных функций. Преимущество позднего связывания – гибкость кода, простота изменения программы; недостаток – уменьшение скорости обработки (построение виртуальных таблиц, определение виртуальных функций).

1. **АЛГОРИТМ**

**АЛГОРИТМ МЕНЮ:**

**ШАГ 0.** Подключение необходимых библиотек

**ШАГ 1. ПОКА (ввод != Enter)**

**ШАГ 1. 1 Инициализировать** объект класса Figure объектом класса Tetrahedron, Cube или Parallelepiped.

**ШАГ 1. 2** Вывести информацию об объекте.

**АЛГОРИТМ СОЗДАНИЯ ОБЪЕКТА:**

**ШАГ 0.** Подключение необходимых библиотек

**ШАГ 1. СОЗДАТЬ** вектор **lol** для хранения чисел**.**

**ШАГ 2. ВВОД** чисел.

**ШАГ 3. ВВОД** в переменную **k**.

**ШАГ 4. ЕСЛИ lol.size() ==** 1

**ЕСЛИ k == 1**

вернуть **Tetrahedron(lol[0])**

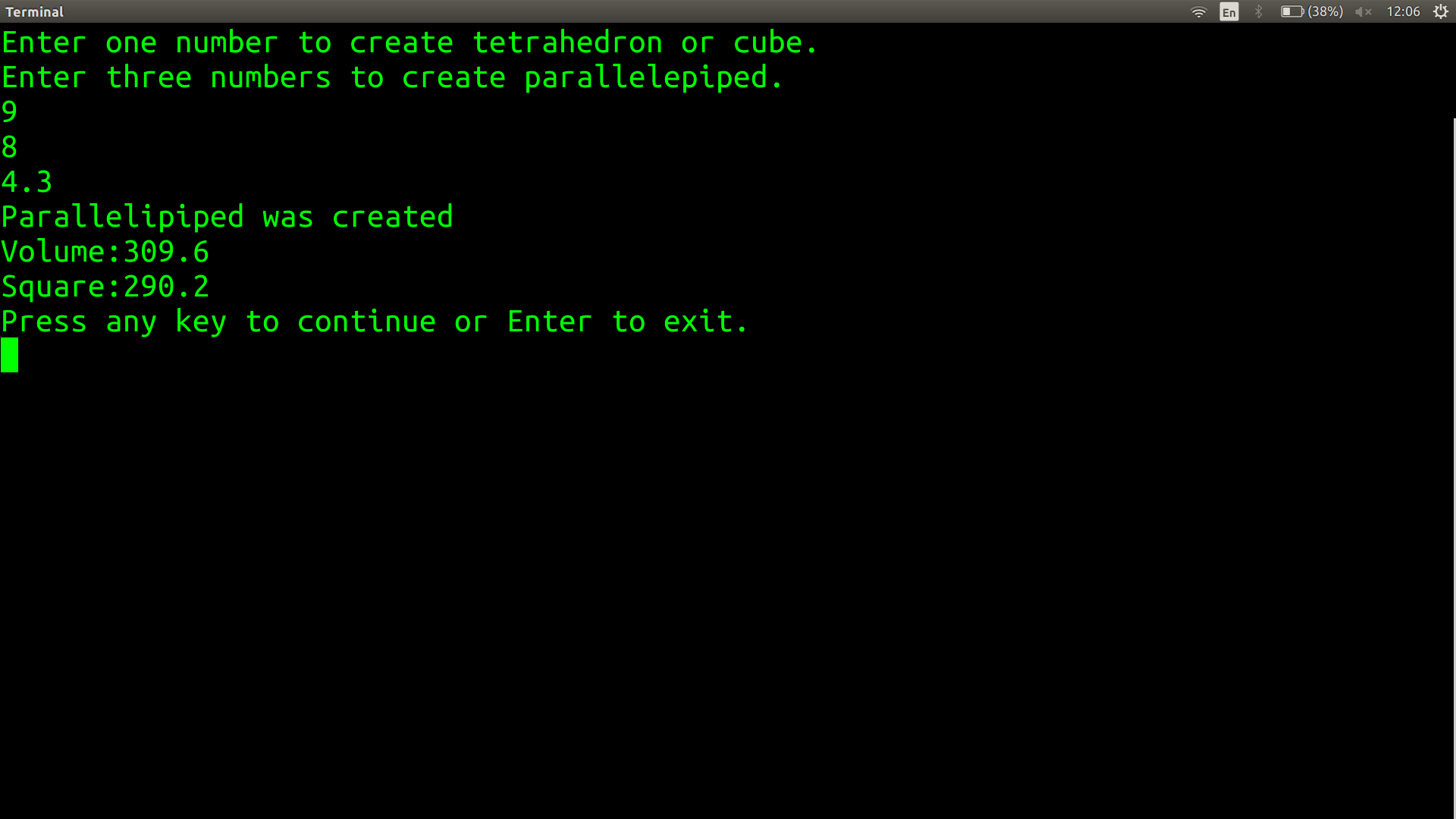
**ЕЩЕ ЕСЛИ k == 2**

вернуть **Cube(lol[0])**

**ЕЩЕ**

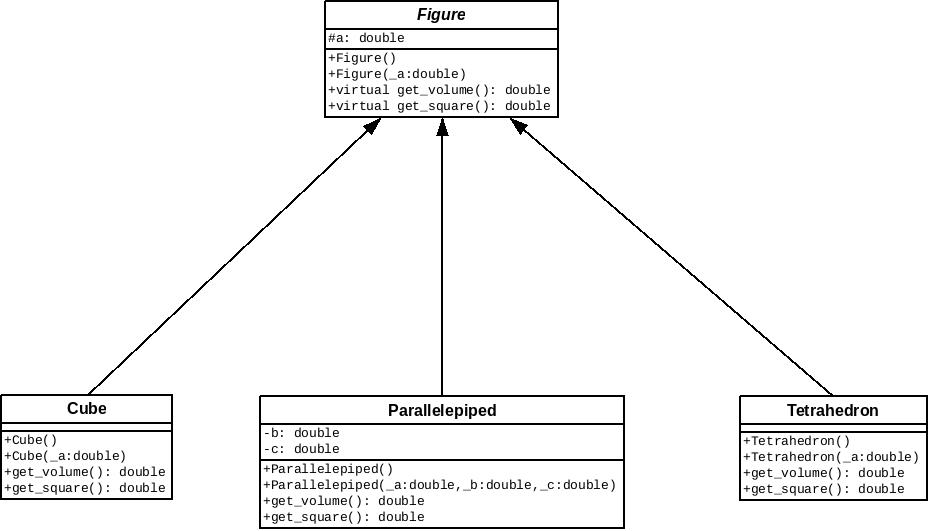
вернуть **Parallelepiped(lol[0], lol[1], lol[2])**

1. **ПРИМЕР РАБОТЫ ПРОГРАММЫ**



**Рисунок 5.1 – пример работы программы**

1. **UML - диаграмма**

**рис. 6.1 – UML-диаграмма**

1. **КОД**

#include <iostream>

#include <vector>

#include <cstring>

#include <string>

#include "figure.h"

#include "tetrahedron.h"

#include "parallelepiped.h"

#include "cube.h"

using namespace std;

void eatline() {

while(getchar() != '\n')

continue;

}

Figure\* init() {

string x;

vector<double> lol;

while (lol.size() != 3) {

getline(cin, x);

if (x.empty()) {

break;

}

else {

double z = stod(x, NULL); // convert string to number

lol.push\_back(z);

}

}

if (lol.size() == 1) {

cout << "Tetrahedron(1) or cube(2)?" << endl;

int sw;

cin >> sw;

eatline();

if (sw == 1) {

cout << "Tetrahedron was created" << endl;

return new Tetrahedron(lol[0]);

}

if (sw == 2) {

cout << "Cube was created" << endl;

return new Cube(lol[0]);

}

}

else {

cout << "Parallelipiped was created" << endl;

return new Parallelepiped(lol[0], lol[1], lol[2]);

}

}

int main() {

cout << "Igor Mayorov, HARDEST LEVEL, var. 8\n"

"Press any key to continue or Enter to exit." << endl;

/\*program is working repeatedly until user presses Enter key\*/

while (getchar() != '\n') {

system("clear");

cout << "Enter one number to create tetrahedron or cube.\n"

"Enter three numbers to create parallelepiped." << endl;

eatline();

Figure\* lol = init();

cout << "Volume:" << lol->*get\_volume*() << endl;

cout << "Square:" << lol->*get\_square*() << endl;

cout << "Press any key to continue or Enter to exit." << endl;

}

return 0;

}

**8. ВЫВОДЫ**

В этой лабораторной работе я изучил навыки работы с наследованием. Научился наследовать закрытые и открытые поля в классах. Изучил, как добавлять новые функции в классе-наследнике, сохраняя при этом все элементы родителей, перегружать и переопределять родительские элементы.