ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | А. Ю. Петров |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| ПЕРЕГРУЗКА ОПЕРАТОРОВ |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1243 |  | 17.05.2024 |  | М. Э. Зайнулин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**1. Описание задания**

Вариант 10

• Унарная операция. Создать класс координаты (x, y, z) с ++ и --.

Операторы: ++ (постфиксная форма, метод) прибавляет 1,5 ко всем координатам,

--(постфиксная форма, метод) вычитает 5,9 от всех координат,

++ (префиксная форма, дружественная функция) складывает все три координаты,

--(префиксная форма, метод) вычитает координаты друг из друга).

• Задание 2. Бинарная операция

Создать класс стек, размер вводится с клавиатуры, заполняется стек случайными числами. Описать операторы сравнения !=, , <=, >=, == для работы с другими экземплярами вашего класса, оператор () возвращает под-стек.

**2. Формализация**

Все данные вводятся с клавиатуры.

• Интерфейс программы реализован командами циклически запрашиваемыми. После вызова команды выполняется соответствующее действие или последовательность действий.

• Программа при запуске запрашивает размер стека. Стек заполняется случайными числами. • Описание команд программы:

Базовые команды стека:

–show – выводит стек на экран;

–pop – вынимает узел из стека;

–push – добавляет узел в стек;

–up – показывает верхний узел стека;

Перегруженные команды для класса Coordinates

obj++ - прибавляет 1,5 ко всем координатам;

obj— - вычитает 5,9 от всех координат;

++obj- складывает все три координаты;

--obj- вычитает координаты друг из друга;

Перегруженные команды для класса Stack

stack\_== - проверят равенство стеков;

stack\_!= - проверят основной стек не равен второстепенному;

stack\_> - проверят основной стек больше второстепенного;

stack\_< - проверят основной стек меньше второстепенного;

stack\_>= - проверят основной стек больше или равен второстепенному;

stack<= - проверят основной стек меньше или равен второстепенному;

Глобальные

--exit – выход из программы;

--change\_st – переключается между основным стеком и второстепенным;;

Динамическое выделение памяти реализовано через new и delete.

• Оба класса (Coordinates и Stack) содержат конструкторы копирования.

• На все попытки ввести некорректные данные программа будет обнулять биты состояния потока ввода (cin.clear()) и игнорировать введенные данные (cin.ignore()), после этого повторно запрашивать данные от пользователя.

• В случае неверно введенной команды программа выведет, что введена неправильная команда.

• Класс Coordinates включает в себя функции:

Основные функции:

getX() – возвращает координату x;

getY() – возвращает координату y;

getZ() – возвращает координату z;

setCoord(float, float, float) – устанавливает координаты x, y, z;

Перегруженные функции:

operator++(int)

operator—(int)

(дружественная функция) float operator++(Coordinates& crd\_1)

operator—()

• Также класс Coordinates включает в себя конструктор для инициализации данных в момент создания объекта. А также деструктор обнуляющий данные (можно исключить так как в данном классе отсутствует работа с динамической памятью).

• Оператор – префиксной формы вычитает y из x, z из y, x из z.

• Все перегруженные функции класса Coordinates за исключение дружественной функции (оператор ++) изменяют данные объекта и возвращают объект.

Перегруженные оператор ++ (префиксный) возвращает результат сложения координат типа float.

Также для удобства работы были перегружены операторы == и !=, проверяющие равенство или не равенство координат.

Класс Stack включает в себя функции:

Основные функции:

popNode() – извлекает узел из стека (освобождая память);

pushNode() – добавляет узел в стек;

getUpData() – возвращает значение верхнего узла стека;

displayStack() – отображает стек на экране;

isEmpty – проверяет пуст стек или нет;

getLinkUp() – возвращает ссылку на верхний элемент стека;

o Перегруженные функции (согласно заданию):

operator== - проверяет равенство стеков по данным и размеру.

operator!= - проверяет неравенство стеков по данным и размеру.

operator> - проверяет больше ли основной стек относительно второстепенного.

operator< - проверяет меньше ли основной стек относительно второстепенного.

operator>= - проверяет больше или равен основной стек относительно второстепенного.

operator<= - проверяет меньше или равен основной стек относительно второстепенного.

operator() – возвращает под-стек.

• Перегруженные функции сравнения == и != проверяет стеки по данным и размеру. Для оператора == возвращаемое значение 2 – значит стеки полностью равны, 1 – значит стеки равны по размеру, 0 – значит стеки не равны. Для оператора != возвращаемое значение 2 – значит, что стеки полностью не равны, 1 – значит одинаковый размер, но разные данные, 0 – значит стеки равны.

• Деструктор класса стек организован через цикл while и выполняется пока стек не пуст. В цикле происходит извлечение узла (popNode), которое и очищает память в куче.

• Для организации стека написана конструктор (stack\_nd) , в котором хранится объекта класса Coordinates и указатель на следующий элемент стека.

• Для структуры stack\_nd написан конструктор обнуляющий указатель на следующий элемент стека.

• В программе используется только два стека для демонстрации работы перегруженных операторов сравнения. Переключение между ними возможно через команду –change\_st.

• Независимо от выбранного стека программа сравнивает основной стек как левый операнд, а вспомогательный как правый операнд

**3. Исходный код программы**

**Файл main.cpp**

#include <string>

#include <limits>

#include "Stack.h"

using namespace std;

int main()

{

    string command;

    Coordinates tmp;

    float tmp\_x = 0.0;

    float tmp\_y = 0.0;

    float tmp\_z = 0.0;

    int stack\_size;

    char stack\_fl = 0x01;

    char fl;

    while (true)

    {

        cout << "Enter size of stack = ";

        if (cin >> stack\_size)

            if (stack\_size > 0)

                break;

        cin.clear();

        cin.ignore(numeric\_limits<std::streamsize>::max(), '\n');

    }

    srand(time(0));

    Stack main\_stack = Stack(stack\_size);

    Stack tmp\_stack = Stack(stack\_size);

    Stack \*stack = &main\_stack;

    while (true)

    {

        cout

        <<"show - Show stack" << endl

        <<"pop  - Popnode from stack"<< endl

        <<"push - Push node to stack"<< endl

        <<"up   - Show item of up node"<< endl

        <<"obj++ - + 1.5 to all coordinates" << endl

        <<"obj-- - + 5.9 from all coordinates" << endl

        <<"++obj - Sum all coordinates" << endl

        <<"--obj - Subtractscoordinates from each other" << endl

        <<"stack\_== - ==" << endl

        <<"stack\_!= - !=" << endl

        <<"stack\_>  - >" << endl

        <<"stack\_<  - <" << endl

        <<"stack\_>= - >=" << endl

        <<"stack\_<= - <=" << endl

        <<"stack\_() - <=" << endl

        <<"exit - Correct exit from program" << endl

        <<"change\_st - Change main stack" << endl

        << "\n";

        cout << "Enter: ";

        cin >> command;

        if (command == "show")

        {

            stack->displayStack();

            continue;

        }

        if (command == "pop")

        {

            stack->popNode();

            continue;

        }

        if (command == "push")

        {

            while (true)

            {

                cout << "Enter coordinates in form x y z: ";

                if (cin >> tmp\_x >> tmp\_y >> tmp\_z)

                {

                    tmp.setCoord(tmp\_x, tmp\_y, tmp\_z);

                    stack->pushNode(tmp);

                    break;

                }

                else

                {

                    cin.clear();

                    cin.ignore(numeric\_limits<std::streamsize>::max(), '\n');

                }

                continue;

            }

        }

        if (command == "up")

        {

            Coordinates &tmp\_link = stack->getLinkUp();

            cout << "Coordinates in form x y z: "

                    << tmp\_link.getX() << "\t"

                    << tmp\_link.getY() << "\t"

                    << tmp\_link.getZ() << endl;

            continue;

        }

        if (command == "obj++")

        {

            stack->getLinkUp()++;

            continue;

        }

        if (command == "obj--")

        {

            stack->getLinkUp()--;

            continue;

        }

        if (command == "++obj")

        {

            cout << "result = " << ++stack->getLinkUp() << endl;

            continue;

        }

        if (command == "--obj")

        {

            --stack->getLinkUp();

            continue;

        }

        if (command == "stack\_==")

        {

            fl = (main\_stack == tmp\_stack);

            if (fl == 2)

                cout << "\tFULL\_EQUAL" << endl;

            else if (fl == 1)

                cout << "\tSIZE\_EQUAL" << endl;

            else if (fl == 0)

                cout << "\tNOT\_EQUAL" << endl;

            continue;

        }

        if (command == "stack\_!=")

        {

            fl = (main\_stack != tmp\_stack);

            if (fl == 2)

                cout << "\tFULL\_NOT\_EQUAL" << endl;

            else if (fl == 1)

                cout << "\tDATA\_NOT\_EQUAL but SIZE\_EQUAL" << endl;

            else if (fl == 0)

                cout << "\tFULL\_EQUAL" << endl;

            continue;

        }

        if (command == "stack\_>")

        {

            fl = (main\_stack > tmp\_stack);

            if (fl == 1)

                cout << "\tMAIN larger than TMP" << endl;

            else

                cout << "\tMAIN not larger than TMP" << endl;

            continue;

        }

        if (command == "stack\_<")

        {

            fl = (main\_stack < tmp\_stack);

            if (fl == 1)

                cout << "\tMAIN lower than TMP\_STACK" << endl;

            else

                cout << "\tMAIN not lower than TMP\_STACK" << endl;

            continue;

        }

        if (command == "stack\_>=")

        {

            fl = (main\_stack >= tmp\_stack);

            if (fl == 1)

            {

                cout << "\tMAIN larger or equal than TMP " << endl;

            }

            else

                cout << "\tMAIN lower than TMP" << endl;

            continue;

        }

        if (command == "stack\_<=")

        {

            fl = (main\_stack <= tmp\_stack);

            if (fl == 1)

            {

                cout << "\tMAIN lower or equal than TMP"

                        << endl;

            }

            else

                cout << "\tMAIN larger than TMP" << endl;

            continue;

        }

        if (command == "stack\_()")

        {

            int skip\_size = 0;

            char fl\_err;

            while (true)

            {

                cout << "Enter num of skip\_nodes = ";

                if (cin >> skip\_size)

                    break;

                else

                {

                    cin.clear();

                    cin.ignore(numeric\_limits<std::streamsize>::max(), '\n');

                }

            }

            Stack sub\_stack = (\*stack)(skip\_size, fl\_err);

            if (fl\_err == true)

                sub\_stack.displayStack();

            continue;

        }

        if (command == "exit")

        {

            break;

        }

        if (command == "change\_st")

        {

            if (stack\_fl == 0x01)

            {

                stack\_fl = 0x02;

                stack = &tmp\_stack;

            }

            else

            {

                stack\_fl = 0x01;

                stack = &main\_stack;

            }

            continue;

        }

    }

    return 0;

}

**Файл Coordinates.cpp**

#include "Coordinates.h"

float operator++(Coordinates& crd\_1);

Coordinates::Coordinates(const Coordinates& obj)

{

    this->x = obj.x;

    this->y = obj.y;

    this->z = obj.z;

}

Coordinates Coordinates::operator++(int)

{

    this->x += 1.5;

    this->y += 1.5;

    this->z += 1.5;

    return Coordinates(\*this);

}

Coordinates Coordinates::operator--(int)

{

    this->x -= 5.9;

    this->y -= 5.9;

    this->z -= 5.9;

    return Coordinates(\*this);

}

Coordinates Coordinates::operator--()

{

    float tmp\_x = this->x,

    tmp\_y = this->y,

    tmp\_z = this->z;

    this->x -= tmp\_y;

    this->y -= tmp\_z;

    this->z -= tmp\_x;

    return Coordinates(\*this);

}

float operator++(Coordinates& crd\_1)

{

    return (crd\_1.x + crd\_1.y + crd\_1.z);

}

bool Coordinates::operator!=(Coordinates& second)

{

    if ((this->getX() != second.getX()) || (this -> getY() != second.getY()) || (this->getZ() != second.getZ()))

    return true;

    return false;

}

bool Coordinates::operator==(Coordinates& second)

{

    if ((this->getX() == second.getX()) && (this -> getY() == second.getY()) && (this->getZ() == second.getZ()))

    return true;

    return false;

}

**Файл Stack.cpp**

#include "Stack.h"

Stack::Stack()

{

    stack\_size = 1;

    up\_node = (stack\_nd) new stack\_node;

    up\_node->next = 0;

}

Stack::Stack(int size)

{

    stack\_size = size;

    up\_node = 0;

    stack\_nd tmp\_dt;

    for (int i = 0; i < stack\_size; ++i)

    {

        tmp\_dt = up\_node;

        up\_node = (stack\_nd) new stack\_node;

        up\_node->data = Coordinates(static\_cast<float>(rand() % 301 - 101) + 0.5, static\_cast<float>(rand() % 301 - 101) + 0.3, static\_cast<float>(rand() % 301 - 101) + 0.1);

        up\_node->next = tmp\_dt;

    }

}

Stack::Stack(const Stack &obj)

{

    this->stack\_size = obj.stack\_size;

    stack\_nd this\_up = (stack\_nd) new stack\_node;

    stack\_nd obj\_up = obj.up\_node;

    this->up\_node = this\_up;

    for (int i = 0; i < stack\_size; ++i)

    {

        this\_up->data = obj\_up->data;

        if (stack\_size != stack\_size - 1)

        {

            this\_up->next = (stack\_nd) new stack\_node;

            this\_up = this\_up->next;

            obj\_up = obj\_up->next;

        }

    }

}

Coordinates Stack::popNode()

{

    if (up\_node == NULL || stack\_size == 0)

    {

        std::cout << "EMPTY\_STACK;" << std::endl;

        return Coordinates();

    }

    Coordinates tmp = up\_node->data;

    stack\_nd dl\_nd = up\_node;

    stack\_size--;

    up\_node = up\_node->next;

    delete dl\_nd;

    return tmp;

}

void Stack::pushNode(Coordinates dt)

{

    stack\_nd tmp = new stack\_node;

    stack\_size++;

    tmp->data = dt;

    tmp->next = up\_node;

    up\_node = tmp;

}

Coordinates &Stack::getUpData()

{

    return (this->up\_node->data);

}

void Stack::displayStack()

{

    if (this->isEmpty())

    {

        std::cout << "EMPTY\_STACK" << std::endl;

        return;

    }

    Coordinates tmp\_data;

    int number = 1;

    std::cout << std::setiosflags(std::ios::left) << std::setfill('=') << std::setw(52) << "=" << std::endl;

    std::cout << std::setfill(' ') << std::setw(14) << "| NUMBER    |" << std::setw(14)<< " " << std::setw(22) << "DATA IN NODE" << "| " << std::endl;

    std::cout << std::setfill('=') << std::setw(52) << "=" << std::endl;

    Stack tmp(\*this);

    while (!tmp.isEmpty())

    {

        tmp\_data = tmp.popNode();

        std::cout << std::setiosflags(std::ios::right) << std::setfill(' ') << "| " << std::setw(9) << number << " |" << std::setiosflags(std::ios::left) << std::setw(4) << " x:" << std::setprecision(4) << std::setw(8) << tmp\_data.getX() << std::setw(4) << "|y: " << std::setprecision(4) << std::setw(8) << tmp\_data.getY() << std::setw(4) << "|z: " << std::setprecision(4) << std::setw(8) << tmp\_data.getZ() << " |" << std::endl;

        std::cout << std::setfill('=') << std::setw(52) << "=" << std::endl;

        number++;

    }

    std::cout.unsetf(std::ios::right);

}

bool Stack::isEmpty()

{

    return (stack\_size == 0) ? 1 : 0;

}

Stack::~Stack()

{

    while (!isEmpty())

        popNode();

}

Stack Stack::operator()(int skip\_nodes, char &fl\_err)

{

    Stack tmp(\*this);

    int i = 0;

    fl\_err = true;

    if (skip\_nodes >= 0 && skip\_nodes < stack\_size)

    {

        while (i != skip\_nodes && !tmp.isEmpty())

        {

            tmp.popNode();

            i++;

        }

    }

    else

    {

        std::cout << "BAD\_NUMBER;\n";

        fl\_err = false;

        return Stack();

    }

    return tmp;

}

Stack Stack::operator=(const Stack &second)

{

    this->~Stack();

    this->stack\_size = second.stack\_size;

    stack\_nd this\_up = (stack\_nd) new stack\_node;

    stack\_nd obj\_up = second.up\_node;

    this->up\_node = this\_up;

    for (int i = 0; i < stack\_size; ++i)

    {

        this\_up->data = obj\_up->data;

        if (stack\_size != stack\_size - 1)

        {

            this\_up->next = (stack\_nd) new stack\_node;

            this\_up = this\_up->next;

            obj\_up = obj\_up->next;

        }

    }

    return \*this;

}

char Stack::operator==(const Stack &second)

{

    Stack first(\*this), second\_tmp(second);

    char res\_fl = 0;

    Coordinates tmp\_1, tmp\_2;

    if (stack\_size == second.stack\_size)

    {

        res\_fl = 2;

        while (!first.isEmpty() && !second\_tmp.isEmpty())

        {

            tmp\_1 = first.popNode();

            tmp\_2 = second\_tmp.popNode();

            if (tmp\_1 != tmp\_2)

            {

                res\_fl = 1;

                break;

            }

        }

    }

    return res\_fl;

}

char Stack::operator!=(const Stack &second)

{

    char tmp\_res = (\*this == second);

    switch (tmp\_res)

    {

    case 1:

        return tmp\_res;

    case 2:

        return false;

    default:

        return 2;

    }

}

bool Stack::operator>(const Stack &second)

{

    if (this->stack\_size > second.stack\_size)

        return true;

    return false;

}

bool Stack::operator<(const Stack &second)

{

    if (this->stack\_size < second.stack\_size)

        return true;

    return false;

}

bool Stack::operator<=(const Stack &second)

{

    if (this->stack\_size <= second.stack\_size)

        return true;

    return false;

}

bool Stack::operator>=(const Stack &second)

{

    if (this->stack\_size >= second.stack\_size)

        return true;

    return false;

}

**Файл Coordinates.h**

#pragma once

#include <iostream>

class Coordinates

{

    private:

    float x;

    float y;

    float z;

    public:

    explicit Coordinates(float X = 0, float Y = 0, float Z = 0) : x(X), y(Y), z(Z) {}

    Coordinates(const Coordinates& obj);

    Coordinates operator++(int);

    Coordinates operator--(int);

    friend float operator++(Coordinates& crd\_1);

    Coordinates operator--();

    bool operator!=(Coordinates& second);

    bool operator==(Coordinates& second);

    float getX()

    {

        return x;

    }

    float getY()

    {

        return y;

    }

    float getZ()

    {

        return z;

    }

    void setCoord(float x, float y, float z)

    {

        this->x = x;

        this->y = y;

        this->z = z;

    }

    ~Coordinates()

    {

        x = 0;

        y = 0;

        z = 0;

    }

};

**Файл Stack.h**

#pragma once

#include "Coordinates.h"

#include <iostream>

#include <iomanip>

#include <ctime>

typedef struct stack\_node

{

    Coordinates data;

    stack\_node\* next;

    stack\_node()

    {

        next = NULL;

    }

} \*stack\_nd;

class Stack

{

    private:

    int stack\_size;

    stack\_nd up\_node;

    public:

    Stack();

    explicit Stack(int size);

    Stack(const Stack& obj);

    Coordinates popNode();

    void pushNode(Coordinates dt);

    Coordinates& getUpData();

    void displayStack();

    bool isEmpty();

    Coordinates& getLinkUp()

    {

        return up\_node->data;

    }

    Stack operator()(int skip\_nodes, char& fl\_err);

    Stack operator=(const Stack& second);

    char operator!=(const Stack& second);

    char operator==(const Stack& second);

    bool operator>(const Stack& second);

    bool operator<(const Stack& second);

    bool operator<=(const Stack& second);

    bool operator>=(const Stack& second);

    ~Stack();

};

**4. Результат работы программы**

На рис. 1 продемонстрировано меню программы

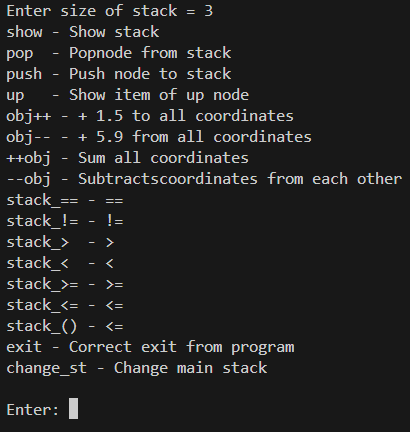


Рисунок 1 – Меню

На рис. 2 продемонстрирована работа команды show, которая выводит стек на экран, где узел под номером 1 самый верхний.

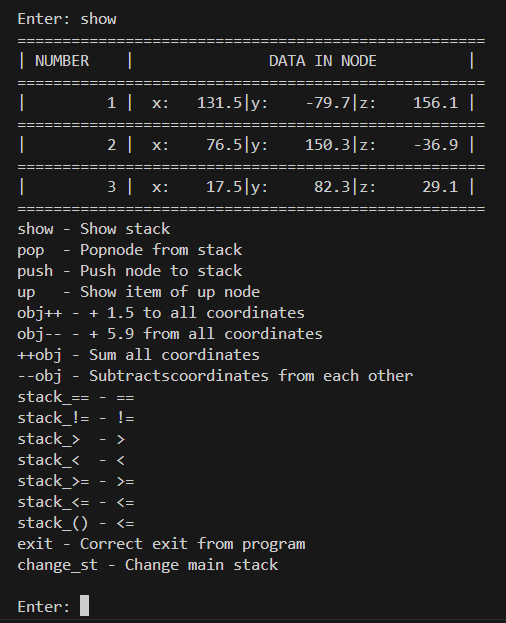


Рисунок 2 – Демонстрация стэка

На рис. 3 продемонстрирована работа команды pop, которая извлекает верхний узел из стека.

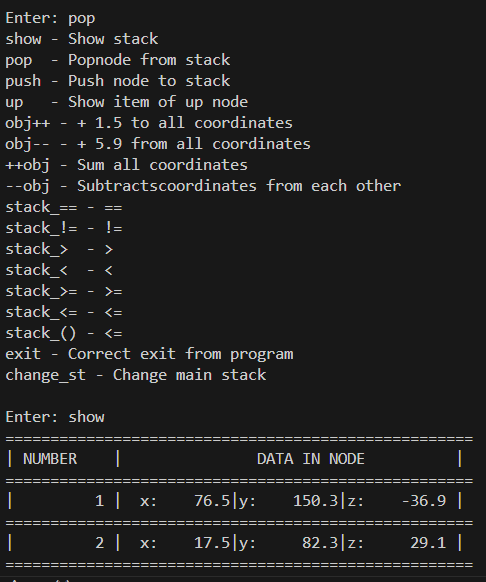


Рисунок 3 – Демонстрация команды pop

На рис. 4 продемонстрирована работа команды push, которая добавляет узел в стек. А также показана корректная обработка неправильно введенных данных.

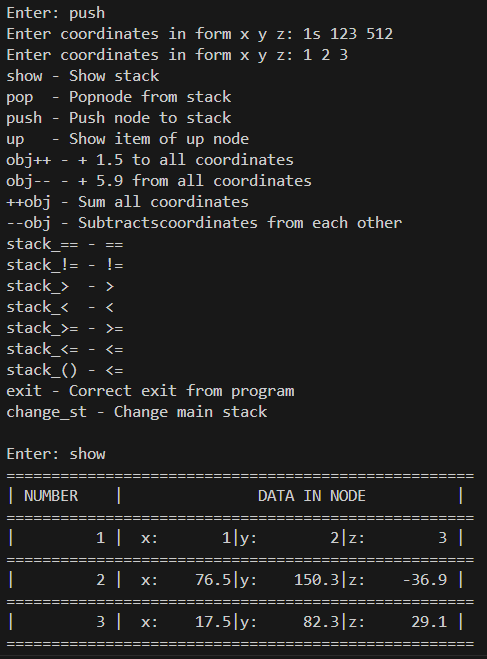


Рисунок 4 – Демонстрация команды push

На рис. 5 продемонстрирована работа команды up, которая показывает данные верхнего узла.

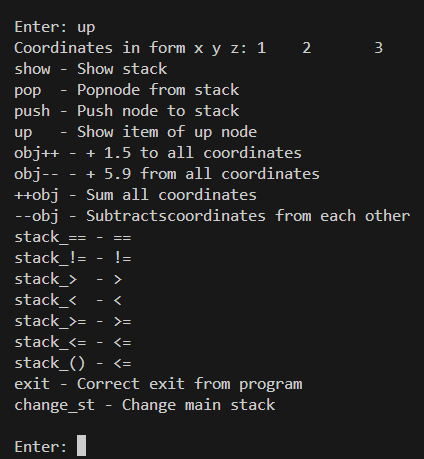
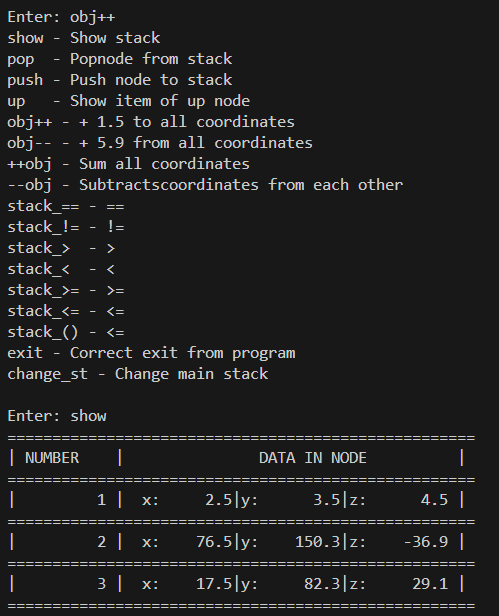
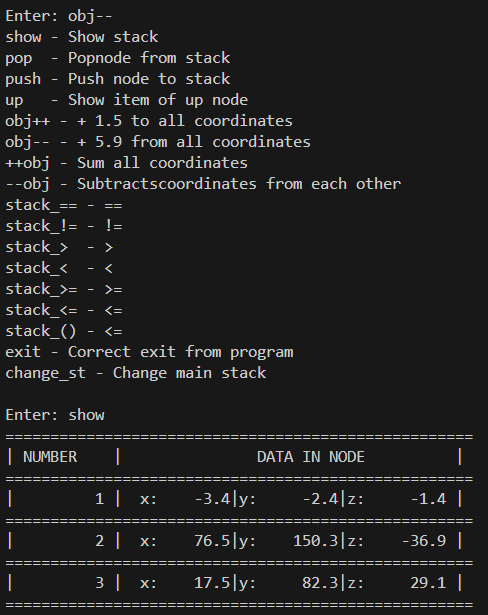


Рисунок 5 – Демонстрация команды up

На рис. 6 продемонстрирована работа команды obj++, которая добавляет 1.5 к каждой координате.

  
Рисунок 6 – Демонстрация команды obj++

На рис. 7 продемонстрирована работа команды obj--, которая вычитает 5.9из каждой координаты.

  
Рисунок 7 – Демонстрация команды obj--

На рис. 8 продемонстрирована работа команды ++obj, которая складывает координаты и выводит их сумму

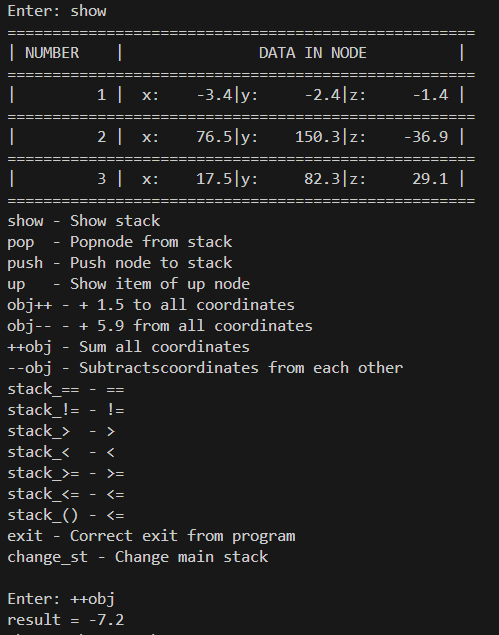


Рисунок 8 – Демонстрация команды ++obj

На рис. 9 продемонстрирована работа команды --obj, которая вычитает y из x, z из y, x из z.

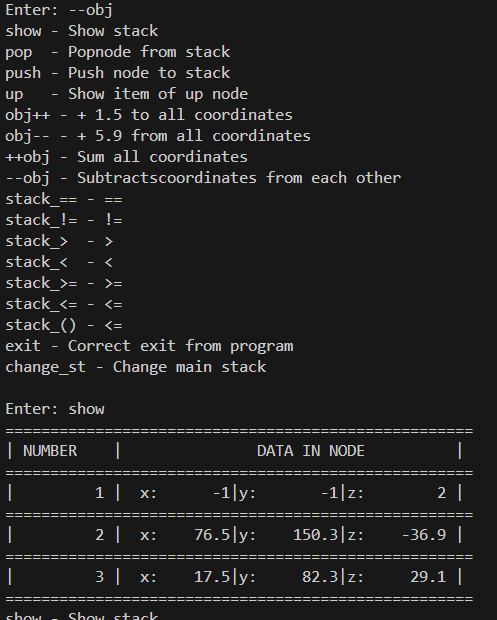


Рисунок 9 – Демонстрация команды –obj

На рис. 10 продемонстрирована работа команды переключения между стеками, а также команда сравнения стека.

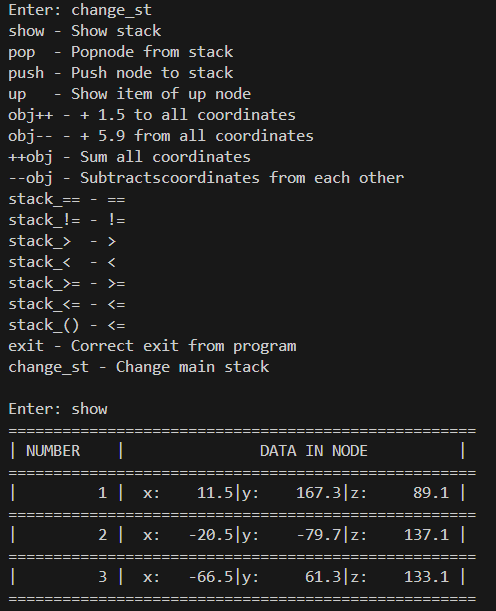


Рисунок 10 – Демонстрация команды change\_st

На рис. 11 продемонстрирована работа команды сравнения стека.

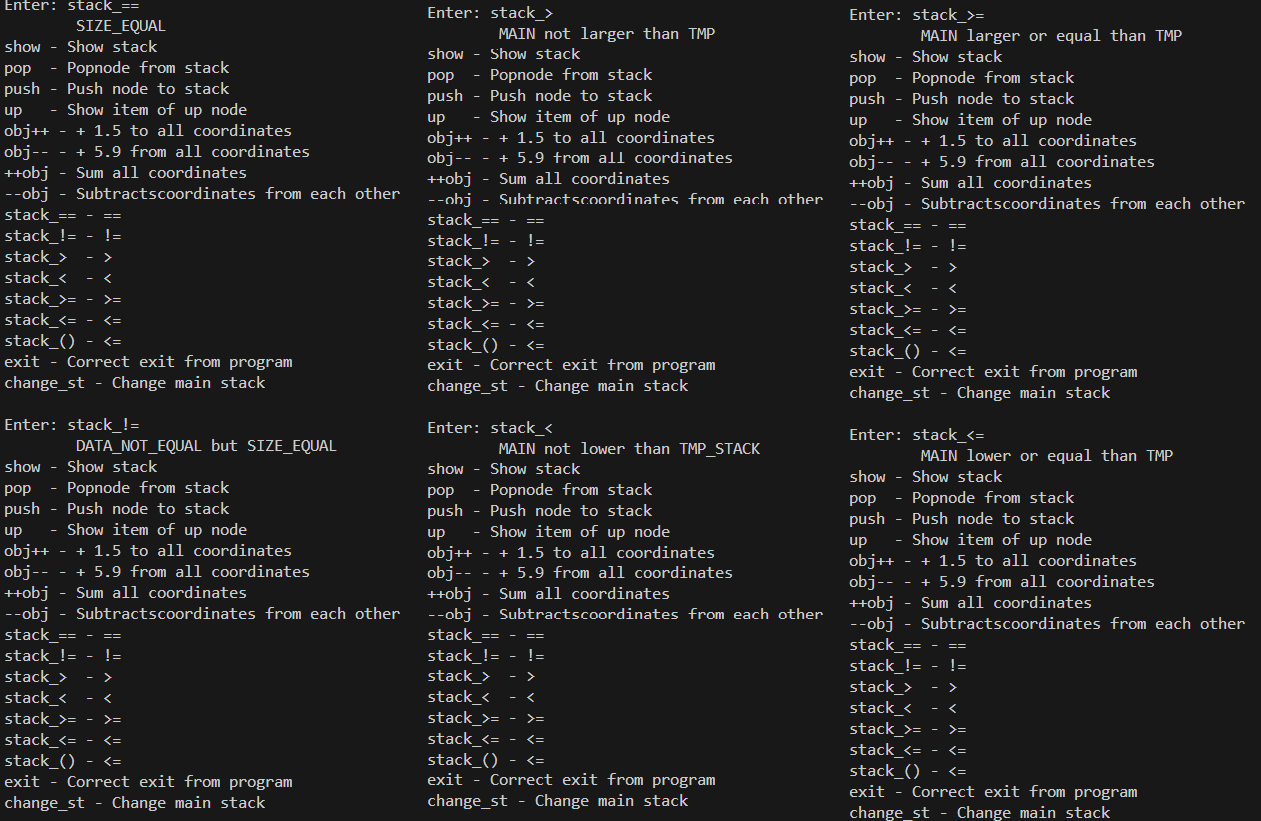


Рисунок 11 – Демонстрация команд сравнения стэков

**5. Выводы**

В данной лабораторной работе требовалось изучить перегрузку унарных и бинарных операторов при помощи методов класса и дружественных функций. При тестировании работы не было выявлено ошибок, программа выдала правильный результат, из чего можно сделать вывод, что лабораторная работа выполнена успешно.