**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Ильиных Вадим Максимович

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

**Вариант №16**

Создать класс **Position** для работы с географическими координатами. Координаты задаются двумя числами широта и долгота. Долгота находится в диапазоне от -180 до 180 градусов. Широта находится в диапазоне от -90 до 90 градусов. Реализовать арифметические операции сложения, вычитания, умножения и деления, а также операции сравнения в виде перегрузки операторов.

Необходимо реализовать пользовательский литерал для работы с константами типа **Position**.

1. **Описание программы**

Создал класс Position для работы с географическими координатами .

Также реализованы операторы -, +, \*, /, =, ==, !=. Оператор вывода реализован как friend-функция, чтобы был доступ к private-переменным класса.

Написан простейший парсер, позволяющий вводить выражения целиком в консоль. Реализовал возможность создавать константные объекта класса при помощи пользовательских литералов

1. **Ссылка на github**

<https://github.com/unchet/oop_exercise_02>

1. **Набор тестов**

Тестовые файлы: **test\_01.txt, test\_02.txt**

**test\_01.txt:**

(1,2)+(3,4)

(48,99)-(-120,-1)

(182,45)\*(3,2)

90,20

13

((12,2)

(-85,80) / (13,76)

(134,-2)/(34,0)

q

**test\_02.txt:**

tgtg

)1,2)+(3,4)

(180,360)=(4,-90)

(45,0)!=(45,0)

(4,-200)+(16,-10)=(20,170)-(0,20)

(1,2)+(3,4)+(-30,-180)-(4,20)

(91,360)=(-89,0)

q

1. **Результаты выполнения тестов**

Все тесты успешно пройдены, программа выдаёт верные результаты

1. **Листинг программы**

**Position.h:**

#ifndef \_\_POSITION\_CLASS\_\_

#define \_\_POSITION\_CLASS\_\_

class Position{

private:

int longitude, latitide;

public:

Position();

Position(int a, int b);

friend std::ostream& operator<<(std::ostream &out, const Position &a);

friend bool Check(Position a);

void operator=(Position a);

Position operator+(Position a);

Position operator-(Position a);

Position operator\*(Position a);

Position operator/(Position a);

bool operator<(Position a);

bool operator<=(Position a);

bool operator>(Position a);

bool operator>=(Position a);

bool operator!=(Position a);

bool operator==(Position a);

};

int operator "" \_lat(unsigned long long a);

int operator "" \_long(unsigned long long a);

Position Parse(const char \*str);

#endif

**Position.cpp:**

#include <iostream>

#include "Position.h"

#include <stack>

Position::Position(){

longitude = 0;

latitide = 0;

}

Position::Position(int a, int b){

latitide = a;

longitude = b;

while(longitude < -180){

longitude += 360;

}

while(longitude > 180){

longitude -= 360;

}

while(latitide < -90 ){

latitide += 180;

}

while(latitide > 90){

latitide -= 180;

}

}

std::ostream& operator<<(std::ostream &out, const Position &a){

out << "(" << a.latitide << "," << a.longitude<< ")" << std::endl;

return out;

}

bool Check(Position a){

return (a.latitide && a.longitude);

}

int operator "" \_lat(unsigned long long a){

return a;

}

int operator "" \_long(unsigned long long a){

return a;

}

void Position::operator=(Position a){

this->longitude = a.longitude;

this->latitide = a.latitide;

}

Position Position::operator+(Position a){

Position res;

res.longitude = this->longitude + a.longitude;

res.latitide = this->latitide + a.latitide;

res = Position(res.latitide, res.longitude);

return res;

}

Position Position::operator-(Position a){

Position res;

res.longitude = this->longitude - a.longitude;

res.latitide = this->latitide - a.latitide;

res = Position(res.latitide, res.longitude);

return res;

}

Position Position::operator\*(Position a){

Position res;

res.longitude = this->longitude \* a.longitude;

res.latitide = this->latitide \* a.latitide;

res = Position(res.latitide, res.longitude);

return res;

}

Position Position::operator/(Position a){

Position res;

res.longitude = this->longitude / a.longitude;

res.latitide = this->latitide / a.latitide;

res = Position(res.latitide, res.longitude);

return res;

}

bool Position::operator!=(Position a){

return(this->longitude != a.longitude || this->latitide != a.latitide);

}

bool Position::operator==(Position a){

return(this->longitude == a.longitude && this->latitide == a.latitide);

}

Position Parse(const char \*str){

int arr[2] = {0};

int cur = 0, minus = 0, brackets = 0, check = 0;

std::stack <char> operations;

std::stack <Position> tmp;

Position a,b;

for(size\_t i = 0; ; ++i){

if((str[i]>=34 && str[i]<=39) || (str[i]>=62 && str[i]!= 58 && str[i]!=59))

break;

if(str[i] == '-' && brackets){

minus = 1;

continue;

}

if(str[i] == '+' || (str[i] == '-' && !brackets)

|| str[i] == '\*' || str[i] == '/' || str[i] == '='

|| str[i] == '!'){

if(!cur)

break;

operations.push(str[i]);

for(int i = 0; i < 2; ++i)

arr[i] = 0;

cur = 0;

minus = 0;

continue;

}

if(str[i] == '('){

check = 1;

++brackets;

continue;

}

if(str[i] == ')'){

if(minus){

arr[cur] \*= -1;

minus = 0;

}

tmp.push(Position(arr[0], arr[1]));

--brackets;

continue;

}

if(str[i] == ','){

if(minus){

arr[cur] \*= -1;

minus = 0;

}

if(cur)

break;

cur++;

continue;

}

if(str[i] == ' ' || str[i] == '\t'){

continue;

}

if(str[i] == '\n'){

if(cur == 1 && !brackets && check){

if(minus)

arr[cur] \*= -1;

while(!operations.empty()){

char c = operations.top();

operations.pop();

switch (c){

case '+':

a = tmp.top();

tmp.pop();

b = tmp.top();

tmp.pop();

tmp.push(a+b);

break;

case '-':

a = tmp.top();

tmp.pop();

b = tmp.top();

tmp.pop();

tmp.push(b-a);

break;

case '\*':

a = tmp.top();

tmp.pop();

b = tmp.top();

tmp.pop();

tmp.push(a\*b);

break;

case '/':

a = tmp.top();

tmp.pop();

b = tmp.top();

tmp.pop();

if(Check(a))

tmp.push(b/a);

else {

std::cout << "Нельзя делить на ноль! Выставлены координаты по умолчанию:" << std::endl;

return Position();

}

break;

}

}

a = tmp.top();

tmp.pop();

return a;

}

else

break;

}

arr[cur] \*= 10;

arr[cur] += str[i] - '0';

}

std::cout << "Неправильная запись! Выставлены координаты по умолчанию:" << std::endl;

return Position();

}

**main.cpp:**

#include <iostream>

#include "Position.h"

int main(){

char c = ' ', r;

char \*str = new char[50];

int comp = 0;

Position a, b;

std::cout << "Это было выведено с помощью пользовательского литерала:" << std::endl;

std::cout << "Широта: " << 12\_lat << std::endl;

std::cout << "Долгота: " << 30\_long << std::endl;

while(1){

std::cout << "Введите выражение: (для выхода введите q)" << std::endl;

for(int i = 0;; ++i){

c = getchar();

if(c == '=' || c == '!'){

comp = 1;

if(c == '!')

getchar();

r = c;

str[i] = '\n';

a = Parse(str);

for(; i >= 0; i--)

str[i]='0';

continue;

}

if(c == 'q'){

delete[] str;

return 0;

}

str[i] = c;

if(c == '\n')

break;

}

b = Parse(str);

if(comp){

switch (r){

case '=':

if(a==b)

std::cout << "Да" << std::endl;

else

std::cout << "Нет" << std::endl;

break;

case '!':

if(a!=b)

std::cout << "Да" << std::endl;

else

std::cout << "Нет" << std::endl;

break;

}

comp = 0;

} else

std::cout << b;

}

}

1. **Вывод**

Реализовал при помощи классов программу на языке С++ для работы с географическими координатами. Используя перегрузку операторов и дружелюбные функции сделал возможным применение к объектам класса операторов сложения, вычитания, умножения, деления, сравнения в соответствии с заданием. Так же добавил возможность создания константных объектов класса при помощи пользовательских литералов. Корректность работы программы проверил на двух тестовых файлах.

1. **Список литературы**

1. Справочник по языку С++ [Электронный ресурс]. URL:<http://cppstudio.com/post/439/> (дата обращения: 3.10.2020)