ФГАОУВО «Южно-Уральский государственный университет (НИУ)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине «Объектно-ориентированное программирование»

Автор работы

студент группы ЕТ-212

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.А. Мухутдинов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Работа зачтена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В. Лут

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Челябинск, 2022

# 1 Постановка задачи

I. Реализовать класс

14. Длина в милях, ярдах, футах, дюймах Length  
1 миля=1760 ярдов  
1 ярд=3 фута=36 дюймов

Конструктор: Length(m[,y[,f[,i]]])

Операции:

x+y, x-y, (увеличить/уменьшить длину на соответствующую длину)

x+=y, x-=y,

z\*x, x/z, (увеличить/уменьшить длину в z раз)

x\*=z, x/=z,

x==y, x!=y, x<y, x>y, x<=y, x>=y,

!x (x равна нулю)

где x,y - длины, z - целое число >=1

вывод, ввод в виде 2m 5y 2f 4i

Методы:

int getMile(); [0..∞]

int getYard(); [0..1759]

int getFoot(); [0..2]

int getInch(); [0..11]

Операции (если есть в задании) =, [], +=, -=, \*=, /=, префиксные ++, -- определять как методы.  
Для ввода переопределить >> , для вывода - << . Формат ввода-вывода объектов делать так, как указано в задании.  
Запись [текст] означает, что текст может отсутствовать, например, конструктор ИмяКласса(a[,b[,c]]) может быть вызван с 1, 2 или 3 аргументами.  
Пример:

Vector a(0,0),b(1.5,0.3);

cout<<"Введите вектор a:"<<endl;

cin>>a;

//нужно ввести (1.2,3.2) вместе со скобками и запятой

cout<<a<<" + "<<b<<" = "<<(a+b)<<endl;

// печатается (1.2,3.2) + (1.5,0.3) = (2.7,3.5)

II. Реализовать main c тестами (создание объектов и выполнение действий с ними)  
III. Написать отчет

* Постановка задачи
* Описание интерфейса класса (class {} и комментарии ко всем полям, методам и функциям)
* Описание тестов для проверки классов (main с комментариями, какие действия выполнялись, полученные результаты)
* Листинг реализации класса (реализация методов и функций, отступы, без комментариев)

# 2 Описание интерфейса класса

class Length{

private:

int inch;//переменная для хранения длины, все хранится в дюймах

public:

//конструктор

Length(int, int, int, int);

//геттеры

int getMile();

int getYard();

int getFoot();

int getInch();

Length &operator+=(const Length &);//увеличить длину на соответствующую длину

Length &operator-=(const Length &);//уменьшить длину на соответствующую длину

Length &operator\*=(int);//увеличить длину в z раз

Length &operator/=(int);//уменьшить длину в z раз

friend bool operator==(const Length &, const Length &);//перегрузка операции ==

friend bool operator!=(const Length &, const Length &);//перегрузка операции !=

friend bool operator<(const Length &, const Length &);//перегрузка операции <

friend bool operator>(const Length &, const Length &);//перегрузка операции >

friend bool operator<=(const Length &, const Length &);//перегрузка операции <=

friend bool operator>=(const Length &, const Length &);//перегрузка операции >=

friend bool operator!(const Length &);//равна ли длина нулю

friend istream& operator>>(istream&, Length&);//перегрузка операции >>

friend ostream& operator<<(ostream&, const Length&);// перегрузка операции <<

};

# 3 Описание тестов для проверки классов

int main(){

Length a(0,0,0,0);

Length b(0,0,0,0);

Length c(0,0,0,0);

cout << "Ввод" << endl;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "Проверка операции !x\n";

cout << " !" << a << " = " << !a << endl;

cout << "Проверка операции x == y\n";

cout << a << " == " << b << " = " << (a == b) << endl;

cout << "Проверка операции x != y\n";

cout << a << " != " << b << " = " << (a != b) << endl;

cout << "Проверка операции x < y\n";

cout << a << " < " << b << " = " << (a < b) << endl;

cout << "Проверка операции x > y\n";

cout << a << " > " << b << " = " << (a > b) << endl;

cout << "Проверка операции x <= y\n";

cout << a << " <= " << b << " = " << (a <= b) << endl;

cout << "Проверка операции x >= y\n";

cout << a << " >= " << b << " = " << (a >= b) << endl;

cout << "Проверка операции x - y\n";

cout << a << " - " << b << " = ";

c = a - b;

cout << c << endl;

cout << "Проверка операции x + y\n";

cout << a << " + " << b << " = ";

c = a + b;

cout << c << endl;

cout << "Проверка операции x += y\n";

cout << a << " += " << b << " = ";

a += b;

cout << a << endl;

cout << "Проверка операции x -= y\n";

cout << a << " -= " << b << " = ";

a -= b;

cout << a << endl;

cout << "Проверка операции z \* y\n";

cout << "5" << " \* " << b << " = ";

c = 5 \* b;

cout << c << endl;

cout << "Проверка операции x / z\n";

cout << a << " / " << "5" << " = ";

c = a / 5;

cout << c << endl;

cout << "Проверка операции y \*= z\n";

cout << b << " \*= " << "9" << " = ";

b \*= 9;

cout << b << endl;

cout << "Проверка операции y \*= z\n";

cout << b << " /= " << "6" << " = ";

b /= 6;

cout << b << endl;

cout << "Вывод" << endl;

cout << "a = " << a << "\nb = " << b << "\nc = " << c;

}

# 4 Листинг реализации класса

Length::Length(int m, int y, int f, int i){inch = m\*1760\*36+y\*36+f\*12+i;}

int Length::getMile(){return inch/(1760\*36);}

int Length::getYard(){return (inch/(3\*12))%1760;}

int Length::getFoot(){return (inch/12)%3;}

int Length::getInch(){return inch%12;}

bool operator==(const Length &L1, const Length &L2){

if(L1.inch == L2.inch) return true;

return false;

}

bool operator!=(const Length &L1, const Length &L2){

if(L1.inch != L2.inch) return true;

return false;

}

bool operator<(const Length &L1, const Length &L2){

if(L1.inch < L2.inch) return true;

return false;

}

bool operator>(const Length &L1, const Length &L2){

if(L1.inch > L2.inch) return true;

return false;

}

bool operator<=(const Length &L1, const Length &L2){

if(L1.inch <= L2.inch) return true;

return false;

}

bool operator>=(const Length &L1, const Length &L2){

if(L1.inch >= L2.inch) return true;

return false;

}

bool operator!(const Length &L){

if(L.inch == 0) return true;

return false;

}

Length &Length::operator+=(const Length &L2){

this->inch += L2.inch;

return \*this;

}

Length &Length::operator-=(const Length &L2){

this->inch -= L2.inch;

return \*this;

}

Length &Length::operator\*=(int z){

this->inch \*= z;

return \*this;

}

Length &Length::operator/=(int z){

this->inch /= z;

return \*this;

}

inline Length operator + (Length &L1, Length &L2){// x + y

int inch = L1.getInch() + L2.getInch();

Length L3(inch/(1760\*36), (inch/(3\*12))%1760, (inch/12)%3, inch%12);

return L3;

}

inline Length operator - (Length &L1, Length &L2){// x - y

int inch = L1.getInch() - L2.getInch();

Length L3(inch/(1760\*36), (inch/(3\*12))%1760, (inch/12)%3, inch%12);

return L3;

}

inline Length operator \* (int z, Length &L2){// z\*x

int inch = z \* L2.getInch();

Length L3(inch/(1760\*36), (inch/(3\*12))%1760, (inch/12)%3, inch%12);

return L3;

}

inline Length operator / (Length &L1, int z){// x/z

int inch = L1.getInch() / z;

Length L3(inch/(1760\*36), (inch/(3\*12))%1760, (inch/12)%3, inch%12);

return L3;

}

istream& operator>>(istream &i, Length &L){

int m, y, f, inch;

i >> m;

i.ignore(1);

i >> y;

i.ignore(1);

i >> f;

i.ignore(1);

i >> inch;

i.ignore(1);

L.inch = m\*1760\*36+y\*36+f\*12+inch;

return i;

}

ostream& operator<<(ostream &o, const Length &L){

return o<<L.inch/(1760\*36)<<"m "<<(L.inch/(3\*12))%1760<<"y "<<(L.inch/12)%3<<"f "<<L.inch%12<<"i";

}