Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «Проектирование программ в ИС»

# Функции-друзья классов. Перегрузка операторов классов

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кабак Д. Н.

**Проверил:**

Монтик Н.С.

Брест 2022

**Цель:** изучить использование friend-функций для доступа к классам извне.

#include <iostream>

using namespace std;

class Student {

private:

string name;

string email;

int age;

int qualification;

public:

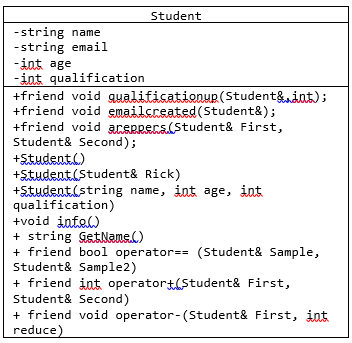
friend void qualificationup(Student&,int);

friend void emailcreated(Student&);

friend void areppers(Student& First, Student& Second);

Student() { name = ""; age = 0; qualification = 0; email = ""; }

Student(Student& Rick) { name = Rick.name; age = Rick.age; qualification = Rick.qualification; email = Rick.email; }

 Student(string name, int age, int qualification) { this->name = name; this->age = age; this->qualification = qualification; }

void info() {

cout << "Info for " << name <<" :" << endl;

cout << "age: " << age << endl;

cout << "qualification: " << qualification << endl;

cout << "email: " << email<<endl<<endl;

}

string GetName() { return name; }

friend bool operator== (Student& Sample, Student& Sample2) {

if (Sample.qualification == Sample2.qualification) {return true;}

else return false;

};

friend int operator+(Student& First, Student& Second) {

return First.qualification + Second.qualification;

};

friend void operator-(Student& First, int reduce) {

First.qualification= First.qualification - reduce;

}

};

void qualificationup(Student& Sample, int upby) { Sample.qualification = Sample.qualification + upby; }

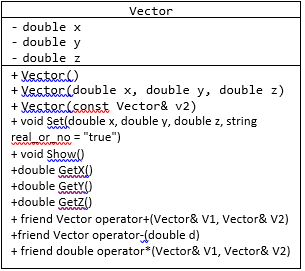
void emailcreated(Student& Sample) { Sample.email = Sample.name + "@mail.ru"; }

void areppers(Student& First, Student& Second) {

if (First.age == Second.age)cout << "the Students are peers";

else cout << "the students aren't peers";

}

class Vector {

private:

double x, y, z;

public:

Vector() {

this->x = 0;

this->y = 0;

this->z = 0;

}

Vector(double x, double y, double z) {

this->x = x;

this->y = y;

this->z = z;

}

Vector(const Vector& v2) {

this->x = v2.x;

this->y = v2.y;

this->z = v2.z;

}

void Set(double x, double y, double z, string real\_or\_no = "true") {

this->x = x;

this->y = y;

this->z = z;

}

void Show() {

cout << "X: " << x << endl;

cout << "Y: " << y << endl;

cout << "Z: " << z << endl;

}

double GetX() { return this->x; }

double GetY() { return this->y; }

double GetZ() { return this->z; }

friend Vector operator+(Vector& V1, Vector& V2) {

return Vector(V1.x + V2.x, V1.y + V2.y, V1.z + V2.z);

}

Vector operator-(double d) {

return Vector(this->x - d, this->y - d, this->z - d);

}

friend double operator\*(Vector& V1, Vector& V2) {

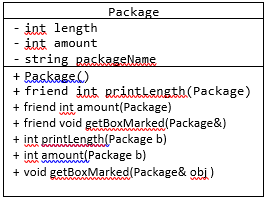
return (V1.x \* V2.x + V1.y \* V2.y + V1.z \* V2.z);

}

};

class Package{

private:

 int length;

int amount;

string packageName;

public:

Package() : length(0) {}

friend int printLength(Package);

friend int amount(Package);

friend void getBoxMarked(Package&);

};

int printLength(Package b){

b.length += 10;

return b.length;

}

int amount(Package b) {

b.amount++;

return b.amount;

}

void getBoxMarked(Package& obj ) {

obj.packageName = obj.packageName + "PinskiyPochtoviyOffice";

}

int main() {

Student Andre("Andre", 19, 3); Student Dandy("Dandy", 18, 5);

Andre.info();

emailcreated(Andre);

Andre - 2;

Andre.info();

Vector arr(1, 4, 5), arr2(6, 7, 8);

Vector arr3 = (arr + arr2);

arr3.Show(); }

Info for Andre :

age: 19

qualification: 3

email:

Info for Andre :

age: 19

qualification: 1

email: Andre@mail.ru

X: 7

Y: 11

Z: 13

**Вывод:** в ходе лабораторной работы я научился использовать friend-функции.