**Приложение 1**

**Текст программы**

#include <iostream>

using namespace std;

const int LEN\_FULL\_NAME = 20,

LEN\_NUMBER = 11;

class Subscriber

{

// Описание полей класса (1)

char \*fullName\_ = new char[LEN\_FULL\_NAME], // ФИО

\*number\_ = new char[LEN\_NUMBER]; // номер телефона

short tariff\_; // Фиксированная величина 1000 руб в месяц

public:

//Конструкторы (2)

Subscriber();

Subscriber(char \*fullName, char \*number, short tariff);

Subscriber(const Subscriber &ob);

//Вспомогательная функция получения данных для объекта

void getob(char \*fullName, char \*number, short tariff);

//Дружественные функции ввода-вывода (3)

friend void in(Subscriber &ob);

friend void out(const Subscriber &ob);

//Перегрузка бинарного оператора + (4)

Subscriber &operator +(short tariff);

// Перегрузка через дружественнную функцию (5)

friend Subscriber &operator -(Subscriber &ob, short tariff); //

//Перегрзука операторов отношения > и <

bool operator >(const Subscriber &ob);

friend bool operator <(const Subscriber &ob1, const Subscriber &ob2);

//Перегрузка оператора = (6)

Subscriber & operator =(const Subscriber &ob);

//Перегрузка постфиксного инкремента(добавляет 100 руб к тарифу) (7.1)

Subscriber &operator ++(int);

//Перегрузка префиксного декремента(снимает 100 руб с тарифа) (7.2)

friend Subscriber &operator --(Subscriber &ob);

//Перегрузка << и >> (8)

friend const ostream& operator << (ostream & os, const Subscriber & ob);

friend istream & operator >> (istream & is, Subscriber & ob);

};

// функция main

#include "Subscriber.h"

Subscriber maxobject(Subscriber \*subscriber, int size);

int main()

{

setlocale(0, "");

try {

int size;

cout << "Введите количество абонентов\n";

cin >> size;

if (!cin) throw "Введено не число\n";

if (size <= 0) throw "число должно быть >=1\n";

Subscriber \*subscriber = new Subscriber[size];

cout << "Введите информацию о абоненте(ах)\n";

// массив из объектов класса

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

cout << "абонент [" << i << "]:" << '\n';

cin >> subscriber[i];

}

cout << "Информация о максимальном cчете\n" << maxobject(subscriber, size);

delete[] subscriber;

}

catch (char\* err){

cerr << err;

}

system ("pause");

}

#include "Subscriber.h"

// Функция, определяющая максимальный объект (10)

Subscriber maxobject(Subscriber \*subscriber, int size) // const и индекс вместо значения max

{

Subscriber maxobject;

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

if (subscriber[i] > maxobject)

maxobject = subscriber[i];

return maxobject;

};

#include "Subscriber.h"

Subscriber::Subscriber() : fullName\_(""), number\_(""), tariff\_(0){}

Subscriber::Subscriber(char \*fullName, char \*number, short tariff) :

fullName\_(fullName), number\_(number), tariff\_(tariff){}

Subscriber::Subscriber(const Subscriber &ob) :

fullName\_(ob.fullName\_), number\_(ob.number\_), tariff\_(ob.tariff\_){};

void Subscriber::getob(char \*fullName, char \*number, short tariff)

{

/\*char \*adc = "123";\*/

fullName\_ = fullName;

//fullName = adc;

number\_ = number;

tariff\_ = tariff;

};

void in(Subscriber &ob)

{

char \*fullName = new char[LEN\_FULL\_NAME],

\*number = new char[LEN\_NUMBER];

int tariff;

cin >> fullName;

cin >> number;

cin >> tariff;

ob.getob(fullName, number, tariff);

}

void out(const Subscriber &ob)

{

cout << "ФИО название: " << ob.fullName\_ << "\n";

cout << "Номер телефона : " << ob.number\_ << "\n";

cout << "Тариф : " << ob.tariff\_ << "\n";

};

Subscriber & Subscriber::operator +(short tariff)

{

tariff\_ += tariff;

tariff\_ %= 1000;

return \*this;

};

Subscriber &operator -( Subscriber &ob, short tariff)

{

ob.tariff\_ -= tariff;

ob.tariff\_ %= 1000;

if (ob.tariff\_<0)

ob.tariff\_ += 1000;

return ob;

};

bool Subscriber::operator >(const Subscriber &ob)

{

return (tariff\_ > ob.tariff\_);

};

bool operator <(const Subscriber &ob1, const Subscriber &ob2)

{

return (ob1.tariff\_ < ob2.tariff\_);

};

Subscriber & Subscriber::operator =(const Subscriber &ob){

if (this!=&ob)

{

fullName\_ = ob.fullName\_;

number\_ = ob.number\_;

tariff\_ = ob.tariff\_;

}

return \*this;

};

Subscriber & Subscriber::operator ++(int)

{

Subscriber temp = \*this;

this -> tariff\_+=100;

if (tariff\_ > 1000)

tariff\_ = 0;

return temp;

};

Subscriber &operator --(Subscriber &ob)

{

if (ob.tariff\_ < 0)

ob.tariff\_ = 1000;

return ob;

}

const ostream& operator << (ostream & os, const Subscriber & ob)

{

os << "ФИО название: " << ob.fullName\_ << "\n"

<< "Номер телефона : " << ob.number\_ << "\n"

<< "Тариф : " << ob.tariff\_ << "\n";

return os;

}

istream & operator >> (istream & is, Subscriber & ob)

{

char \*fullName = new char[LEN\_FULL\_NAME],

\*number = new char[LEN\_NUMBER];

short tariff;

is >> fullName;

is >> number;

is >> tariff;

ob.getob(fullName, number, tariff);

return is;

}