**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИТЕТ**

**им. И.Раззакова**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «**Программное обеспечение компьютерных систем**»

Направление: 710400 «**Программная инженерия**»

Дисциплина: «**Объектно-ориентированное программирование**»

ОТЧЕТ

Лабораторная работа №3

Тема: “ **Конструктор и деструктор**”

Выполнила: Бактыбекова Айтунук

Группа: ПИ-2-21

Проверил: Мусабаев Э.Б.

Бишкек – 2024

Задание №1

Постановка задачи:

1. Разработайте программу с классом **Timer**, в которой таймер действует как таймер обратного отсчета. При создании объекта типа **timer** ему присваивается начальное значение времени. В результате вызова функции **run()** таймер начинает отсчет в сторону уменьшающихся значений, пока не достигнет значения 0, после чего зазвонит звонок. **Создайте 3 варианта конструктора, т.е. конструктор перегрузите 3 раза** для того, чтобы можно было указывать время в секундах с помощью целого числа или строки, или в минутах и секундах, если указываются два целых числа.

Эта программа использует библиотечную функцию **clock(),** возвращающую число тиков, прошедших с момента запуска программы. Поделив это значение на макрос **CLK\_TCK**, получаем значение в секундах. Прототипы для **clock**() и **CLK\_TCK** содержатся в заголовочном файле time.h.

Класс **Timer**():

class Timer

{

private:

int minutes;

int seconds;

В методе **run()** класса **Timer** реализуется таймер, который отсчитывает время в обратном порядке от указанных минут и секунд до нуля. Во время работы этот метод выводит текущее время в формате "мм:сс" с использованием манипуляторов **setw** и **setfill** для форматирования вывода.

Созданы:

Timer() // конструктор по умолчанию

{

minutes = 0;

seconds = 3;

}

Timer(int second) { // конструктор для целочисленных данных

minutes = 0;

seconds = second;

}

Timer(const char\* second) { // конструктор для символьных данных

minutes = 0;

seconds = stoi(second);

}

Timer(int minute, int second) { // конструктор для двух аргументов, в мин и сек.

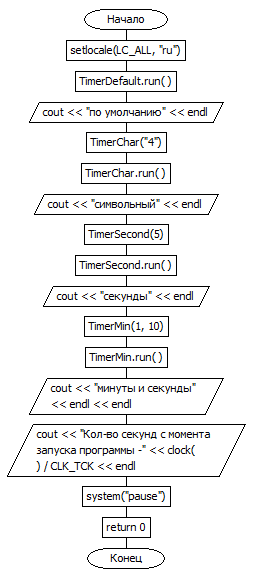
minutes = minute;

seconds = second;

}

В функции **main()** создаются объекты класса **Timer** с разными параметрами и вызывается метод **run()** для каждого из них. Затем выводится количество секунд с момента запуска программы, используя функцию **clock()**.

Блок-схема:



**Код программы:**

#include <iostream>

#include <iomanip>

#include <time.h>

#include <string>

#include <stdio.h>

using namespace std;

class Timer

{

private:

int minutes;

int seconds;

public:

void run()

{

clock\_t t1, t2;

t1 = t2 = clock() / CLK\_TCK;

while (minutes || seconds) {

if (t1 / CLK\_TCK + 1 <= (t2 = clock()) / CLK\_TCK) {

cout << setw(2) << setfill('0') << minutes << ":" << setw(2) << setfill('0') << seconds << endl;

if (seconds == 0) {

minutes--;

seconds = 59;

}

else {

seconds--;

}

t1 = t2;

}

}

cout << "Мелодия \a" << endl; // звонок

}

Timer() // конструктор по умолчанию

{

minutes = 0;

seconds = 3;

}

Timer(int second) { // конструктор для целочисленных данных

minutes = 0;

seconds = second;

}

Timer(const char\* second) { // конструктор для символьных данных

minutes = 0;

seconds = stoi(second);

}

Timer(int minute, int second) { // конструктор для двух аргументов, в мин и сек.

minutes = minute;

seconds = second;

}

};

int main()

{

setlocale(LC\_ALL, "ru");

Timer TimerDefault;

TimerDefault.run();

cout << "по умолчанию" << endl;

Timer TimerChar("4");

TimerChar.run();

cout << "символьный" << endl;

Timer TimerSecond(5);

TimerSecond.run();

cout << "секунды" << endl;

Timer TimerMin(1, 10);

TimerMin.run();

cout << "минуты и секунды" << endl << endl;

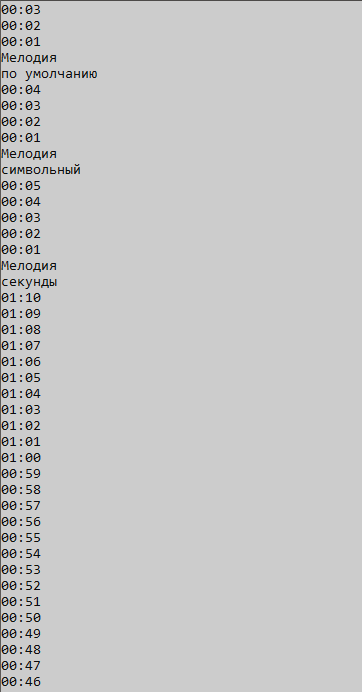
cout << "Кол-во секунд с момента запуска программы -" << clock() / CLK\_TCK << endl;

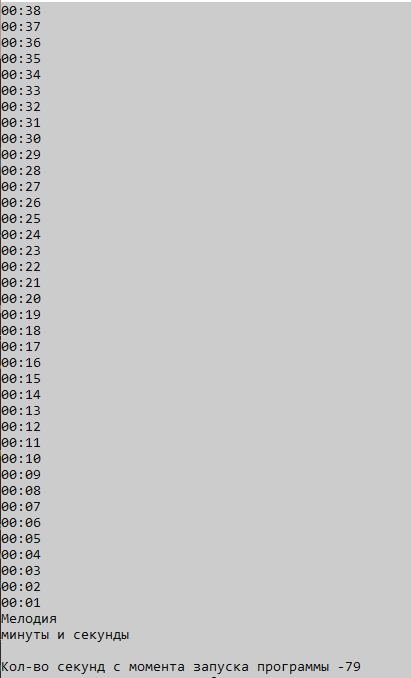
system("pause");

return 0;

}

Результат:





Задание №2

Постановка задачи:

Создайте класс **Nomenclature**, описывающий товары на складе магазина. Закрытыми элементами класса будут: название товара, оптовая цена, розничная наценка и количество товаров на складе. Включите в класс открытые функции подсчета возможного чистого дохода при продаже этого товара и вывода всех данных о товаре на экран. Для инициализации и удаления объектов класса используйте конструкторы и деструкторы.

Описание исходных данных:

Класс **Nomenclature:**

* Описывает товары с их названием, оптовой ценой, розничной наценкой и количеством товара.
* Включает конструктор, который инициализирует объекты класса с заданными значениями.
* Включает метод **getCountProfit()**, который возвращает прибыль от продажи товара.
* Включает методы для установки и получения значений полей объекта.
* Включает пустой деструктор, так как в данном случае нет необходимости освобождать какие-либо ресурсы.

Функции **readProduct и printProduct:**

**readProduct** используется для ввода данных о товаре с клавиатуры и их установки для объекта **Nomenclature.**

**printProduct** выводит информацию о товаре на экран.

Функция **findByProductName:**

Ищет товар в векторе products по его названию и возвращает указатель на него.

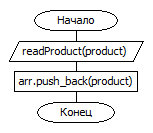
Функции **addProduct и delProduct:**

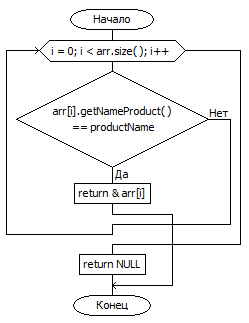
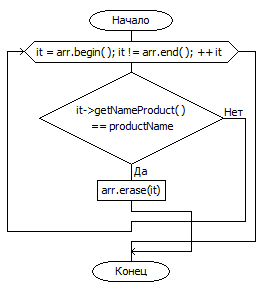
**addProduct** добавляет новый товар в вектор products.

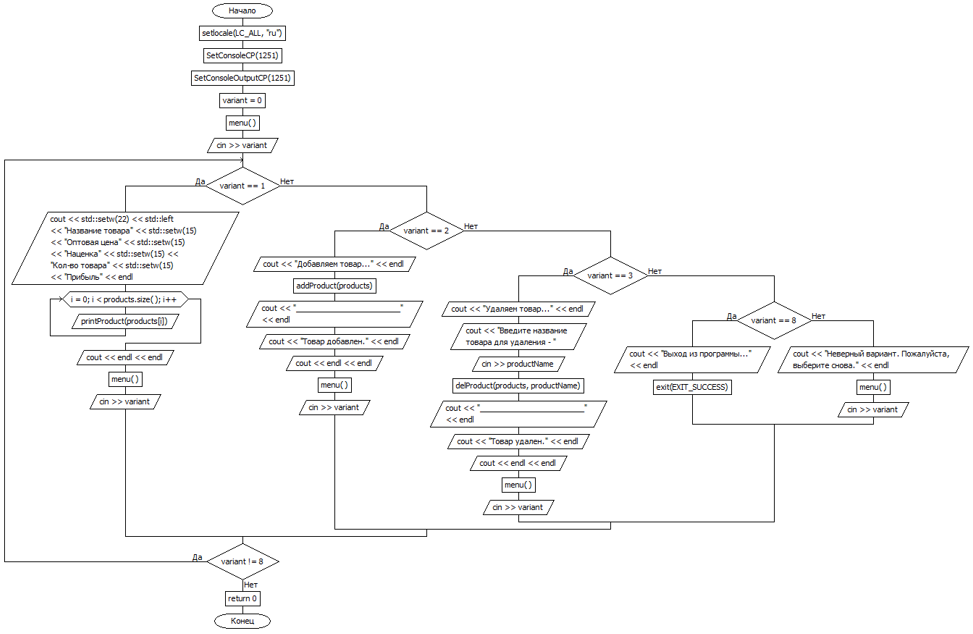
**delProduct** удаляет товар из вектора products по его названию.

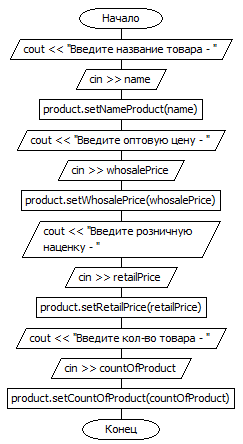
В **main** функции создается вектор **products**, содержащий объекты класса **Nomenclature**.

Блок-схема:









Код программы:

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <Windows.h>

#include <string>

#include <vector>

using namespace std;

class Nomenclature {

private:

string nameProduct;

int whosalePrice;

int retailPrice;

int countOfProduct;

public:

Nomenclature() {}

Nomenclature(const string& nameProduct, int whosalePrice, int retailPrice, int countOfProduct)

: nameProduct(nameProduct), whosalePrice(whosalePrice), retailPrice(retailPrice), countOfProduct(countOfProduct) {}

~Nomenclature() {} // деструктор

int getCountProfit() const {

int outlay = countOfProduct \* whosalePrice;

int profit = ((whosalePrice + retailPrice) \* countOfProduct) - outlay;

return profit;

}

void setNameProduct(const string& name) {

this->nameProduct = name;

}

string getNameProduct() const {

return nameProduct;

}

void setWhosalePrice(int whosalePrice) {

this->whosalePrice = whosalePrice;

}

int getWhosalePrice() const {

return whosalePrice;

}

void setRetailPrice(int retailPrice) {

this->retailPrice = retailPrice;

}

int getRetailPrice() const {

return retailPrice;

}

void setCountOfProduct(int countOfProduct) {

this->countOfProduct = countOfProduct;

}

int getCountOfProduct() const {

return countOfProduct;

}

};

void readProduct(Nomenclature& product) {

string name;

int whosalePrice, retailPrice, countOfProduct;

cout << "Введите название товара - ";

cin >> name;

product.setNameProduct(name);

cout << "Введите оптовую цену - ";

cin >> whosalePrice;

product.setWhosalePrice(whosalePrice);

cout << "Введите розничную наценку - ";

cin >> retailPrice;

product.setRetailPrice(retailPrice);

cout << "Введите кол-во товара - ";

cin >> countOfProduct;

product.setCountOfProduct(countOfProduct);

}

void printProduct(const Nomenclature& product) {

cout << std::setw(22) << std::left << product.getNameProduct()

<< setw(15) << product.getWhosalePrice()

<< setw(15) << product.getRetailPrice()

<< setw(15) << product.getCountOfProduct()

<< setw(15) << product.getCountProfit() << endl;

}

Nomenclature\* findByProductName(vector<Nomenclature>& arr, const string& productName) {

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

if (arr[i].getNameProduct() == productName) {

return &arr[i];

}

}

return NULL;

}

void addProduct(vector<Nomenclature>& arr) {

Nomenclature product;

readProduct(product);

arr.push\_back(product);

}

void delProduct(vector<Nomenclature>& arr, const string& productName) {

for (auto it = arr.begin(); it != arr.end(); ++it) {

if (it->getNameProduct() == productName) {

arr.erase(it);

break;

}

}

}

void menu() {

cout << "Выберите вариант:\n" << endl;

cout << "1. Вывести все товары\n"

<< "2. Добавить новый товар\n"

<< "3. Удалить товар\n"

<< "8. Выход\n" << endl;

cout << ">>> ";

}

int main() {

setlocale(LC\_ALL, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<Nomenclature> products;

int variant = 0;

menu();

cin >> variant;

do {

if (variant == 1) {

cout << std::setw(22) << std::left << "Название товара"

<< std::setw(15) << "Оптовая цена"

<< std::setw(15) << "Наценка"

<< std::setw(15) << "Кол-во товара"

<< std::setw(15) << "Прибыль" << endl;

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

printProduct(products[i]);

}

cout << endl << endl;

menu();

cin >> variant;

}

else if (variant == 2) {

cout << "Добавляем товар..." << endl;

addProduct(products);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Товар добавлен." << endl;

cout << endl << endl;

menu();

cin >> variant;

}

else if (variant == 3) {

cout << "Удаляем товар..." << endl;

string productName;

cout << "Введите название товара для удаления - ";

cin >> productName;

delProduct(products, productName);

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Товар удален." << endl;

cout << endl << endl;

menu();

cin >> variant;

}

else if (variant == 8) {

cout << "Выход из программы..." << endl;

exit(EXIT\_SUCCESS);

}

else {

cout << "Неверный вариант. Пожалуйста, выберите снова." << endl;

menu();

cin >> variant;

}

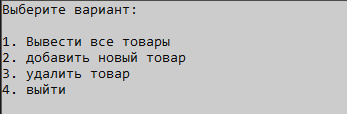
} while (variant != 8);

return 0;

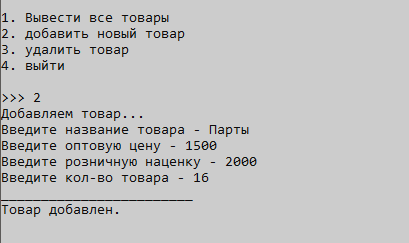
}

**Результат:**

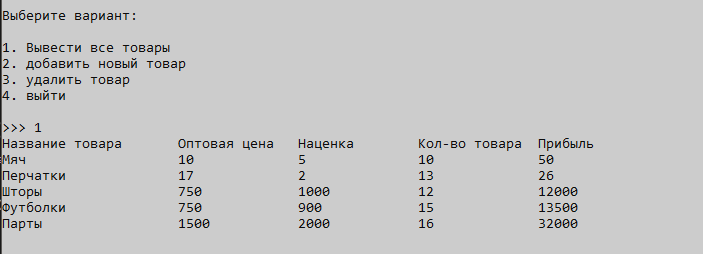
**Меню**



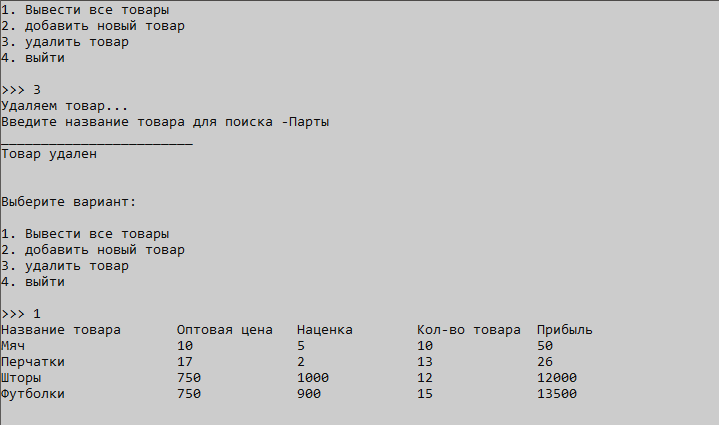
**Добавление товара**



**Просмотр товаров**



**Удаление товара**



Задание №3

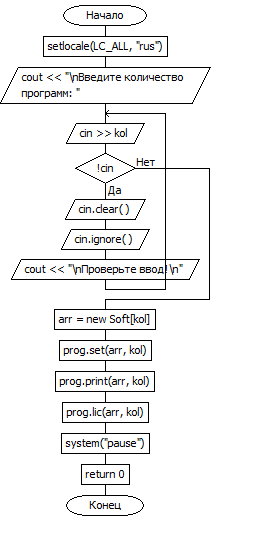
Постановка задачи:

Создайте класс **Soft**, который содержит информацию об установленном программном обеспечении. Закрытыми элементами класса будут: название программы, разработчик, занимаемый объем, дата завершения лицензии. Включите в класс открытые функции подсчета количества дней до завершения лицензии и вывода всех данных об установленном программном обеспечении на экран. Для инициализации и удаления объектов класса используйте конструкторы и деструкторы.

1. **Метод set:**
   * Метод **set** заполняет массив объектов класса **Soft** информацией о программном обеспечении.
   * Для каждого объекта запрашивается название программы, разработчик, объем памяти и дата окончания лицензии.
2. **Метод print:**
   * Метод **print** выводит информацию о программном обеспечении на экран.
   * Для каждого объекта выводится его название, разработчик, объем памяти и дата окончания лицензии.
3. **Метод lic:**
   * Метод **lic** подсчитывает количество оставшихся дней, месяцев и лет лицензии для каждой программы.
   * Сначала запрашивается сегодняшняя дата, а затем для каждой программы вычисляется оставшееся время лицензии.
   * Если лицензия истекла, выводится сообщение о том, что лицензия закончилась.
   * Иначе выводится количество оставшихся дней, месяцев и лет.
4. **Метод main:**
   * В функции **main** создается объект класса **Soft**.
   * Затем запрашивается количество программ и создается массив объектов **Soft**.
   * Вызываются методы **set**, **print** и **lic** для заполнения, вывода информации о программном обеспечении и подсчета оставшегося времени лицензии.
   * После выполнения программы освобождается динамическая память, выделенная для массива объектов **Soft**.
5. **Конструктор по умолчанию (Soft())**:
   * Этот конструктор инициализирует объект класса **Soft** с заданными значениями по умолчанию.
   * Поля **name\_prog**, **name\_razrab**, **memory** инициализируются строкой "0".
   * Поля **day\_lic**, **month\_lic** и **year\_lic** инициализируются нулями.
6. **Деструктор (~Soft())**:
   * Деструктор класса вызывается автоматически при удалении объекта класса, например, при выходе из области видимости, в которой был создан объект, или при вызове оператора **delete** для объекта, созданного в динамической памяти.
   * В данном случае, деструктор **~Soft()** не содержит каких-либо специфических действий, поскольку нет необходимости освобождать какие-либо ресурсы, выделенные в динамической памяти или выполнять другие операции при уничтожении объекта класса **Soft**.

Таким образом, конструктор и деструктор предоставляют базовое начальное и завершающее состояния для объектов класса **Soft**.

Блок-схема:



Код программы:

#include<iostream>

#include<iomanip>

using namespace std;

class Soft {

string name\_prog;

string name\_razrab;

string memory;

int day\_lic;

int month\_lic;

int year\_lic;

public:

Soft() : name\_prog("0"), name\_razrab("0"), memory("0"), day\_lic(0), month\_lic(0), year\_lic(0)

{

}

~Soft()

{

}

void set(Soft\* arr, int& kol) {

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

cout << "\nЗаполните информацию по программе №" << i + 1 << ":" << endl;

cout << "Название программы: ";

cin >> arr[i].name\_prog;

cout << "Наименование разработчика: ";

cin >> arr[i].name\_razrab;

cout << "Объём памяти: ";

cin >> arr[i].memory;

cout << "Дата окончания лиценции (вводится через пробелы, по типу DD\_MM\_YYYY): ";

cin >> arr[i].day\_lic >> arr[i].month\_lic >> arr[i].year\_lic;

}

}

void print(Soft\* arr, int& kol) {

cout << "\n[Информация по программам]\n" << endl;

cout << setiosflags(ios::left) << setw(25) << "[Название программы]" << setw(20) << "[Разработчик]" << setw(25) << "[Занимаемая память]" << setw(25) << "[Дата конца лицении]" << endl << endl;

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

cout << setiosflags(ios::left) << setw(26) << arr[i].name\_prog << setw(20) << arr[i].name\_razrab << setw(25) << arr[i].memory << setw(3) << arr[i].day\_lic << setw(3) << arr[i].month\_lic << setw(7) << arr[i].year\_lic << endl << endl;

}

}

void lic(Soft\* arr, int& kol) {

int today\_day, today\_month, today\_year, licen\_day, licen\_month, licen\_year;

cout << "\n[Информация по количеству дней оставшейся лицензии]\n" << endl;

cout << "\nВведите сегодняшнюю дату (через пробелы, по типу: DD MM YYYY): ";

cin >> today\_day >> today\_month >> today\_year;

cout << "\n[Подсчёт оставшихся дней лицензии]\n" << endl;

cout << setiosflags(ios::left) << setw(25) << "Название программы:" << setw(40) << "Лицензия (осталось времени):" << endl;

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

licen\_year = arr[i].year\_lic - today\_year;

licen\_month = arr[i].month\_lic - today\_month;

licen\_day = arr[i].day\_lic - today\_day;

if (licen\_day < 0) {

licen\_month--;

licen\_day += 30; // Предполагаем, что в месяце 30 дней

}

if (licen\_month < 0) {

licen\_year--;

licen\_month += 12; // В году 12 месяцев

}

if (licen\_year < 0) {

cout << setiosflags(ios::left) << setw(25) << arr[i].name\_prog << "Лицензия закончилась!" << endl;

}

else {

cout << setiosflags(ios::left) << setw(25) << arr[i].name\_prog

<< licen\_day << " дней "

<< licen\_month << " месяцев "

<< licen\_year << " лет" << endl;

}

}

}

};

int main() {

setlocale(LC\_ALL, "rus");

Soft prog;

int kol;

cout << "\nВведите количество программ: ";

metka:cin >> kol;

if (!cin) {

cin.clear();

cin.ignore();

cout << "\nПроверьте ввод!\n";

goto metka;

}

Soft\* arr = new Soft[kol];

prog.set(arr, kol);

prog.print(arr, kol);

prog.lic(arr, kol);

delete[] arr;

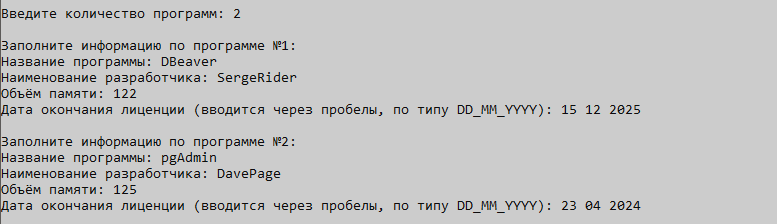
system("pause");

return 0;

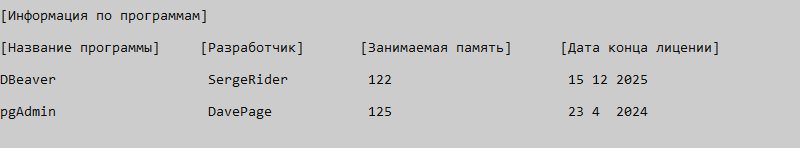
}

Результат:

**Добавление программ**



**Информация по программам**



**Информация по программам (подсчет оставшихся дней лицензии)**

