**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»**

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра программных систем

Дисциплина

**Объектно-ориентированное программирование**

**ОТЧЕТ**по лабораторной работе № 4  
 **Объектно-ориентированное программирование на языке С++.**

Вариант № 3

Студент: Боряков Никита  
Группа: 6102-020302D  
  
Преподаватель: Баландин А.В.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2022

**Задание**

Написать программу управления информацией, состоящей из трех списков: 1) Динамически формируемый список датчиков формата: {; ;}. 2) Динамически формируемый список подсистем: {}. 3) Динамически формируемый список индексов формата: {; }. Список индексных записей фиксирует принадлежность датчиков подсистемам. Датчик может быть задействован только в одной подсистеме или нигде. Индекс записи в списке датчиков – это порядковый номер записи. Индекс записи в списке подсистем – это указатель записи в списке. Шифры датчиков – трёхзначные целые числа. Названия датчиков – символьные строки переменной длины. Шифры подсистем - двузначные целые числа. Названия подсистем – символьные строки произвольной длины. Программа должна обеспечивать выполнение следующих функций:

1. Добавление нового датчика (добавление записи в конец списка датчиков).

2. Добавление новой подсистемы (добавление записи в конец списка подсистем).

3. Включение датчика в подсистему.

4. Исключение датчика из подсистемы.

5. Удаление подсистемы (удаление записи из списка подсистем).

6. Удаление записей неиспользуемых датчиков из списка датчиков.

7. Вывод на экран по заданному шифру подсистемы списка входящих в неё датчиков.

**Информационная модель предметной области**

1. **Информационный объект «База данных»**

Свойства:

* Список записей спецификаций датчиков;
* Список записей спецификаций подсистем;
* Список связей между датчиками и подсистемами.

Процедуры:

* Добавление датчика;
* Добавление подсистемы;
* Включение датчика в подсистему;
* Исключение датчика из подсистемы;
* Удаление записи подсистемы из списка подсистем;
* Удаление записей неиспользуемых датчиков из списка датчиков;
* Вывод по шифру подсистемы входящих в неё датчиков;
* Вывод списка датчиков;
* Вывод списка подсистем;
* Удаление записей в списках.

1. **Информационный объект «Запись спецификации датчиков»**

Свойства:

* Шифр датчика;
* Название датчика.

1. **Информационный объект «Запись спецификации подсистем»**

Свойства:

* Шифр подсистемы;
* Название подсистемы.

1. **Информационный объект «Индексные записи»**

Свойства:

* Индекс датчика в файле SENSOR.DAT;
* Индекс подсистемы в файле SUBSYS.DAT.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ (OOD)

1. **Информационный объект «База данных» (класс «Database»)**

Свойства:

* Список записей спецификаций датчиков (sensor\*);
* Список записей спецификаций подсистем (subsystem\*);
* Список связей между датчиками и подсистемами (link\*).

Процедуры:

* Добавление датчика (метод-add\_sensor());
* Добавление подсистемы (метод-add\_subsystem());
* Включение датчика в подсистему (метод-bind());
* Исключение датчика из подсистемы (метод-unbind());
* Удаление записи подсистемы из списка подсистем (метод-delete\_subsystem());
* Удаление записей неиспользуемых датчиков из списка датчиков (метод-delete\_unused());
* Вывод по шифру подсистемы входящих в неё датчиков (метод-print\_sensors\_by\_subsystem());
* Вывод списка датчиков (метод-print\_sensors());
* Вывод списка подсистем (метод-print\_subsystems());
* Удаление записей в списках (деструктор).

1. **Информационный объект «Датчик»**

Свойства:

* Шифр датчика (unsigned);
* Название датчика (string).

1. **Информационный объект «Подсистема»**

Свойства:

* Шифр подсистемы (unsigned);
* Название подсистемы (string).

1. **Информационный объект «Индексный файл»**

Свойства:

* Индекс датчика в файле SENSOR.DAT (sensor\*);
* Индекс подсистемы в файле SUBSYS.DAT (subsystem\*).

**Структура алгоритма программы**

* Добавление датчика;
* Добавление подсистемы;
* Включение датчика в подсистему;
* Исключение датчика из подсистемы;
* Удаление записи подсистемы из списка подсистем;
* Удаление записей неиспользуемых датчиков из списка датчиков;
* Вывод по шифру подсистемы входящих в неё датчиков;
* Удаление записей в списках.

**Код программы**

#pragma once

#include <list>

#include <string>

#include <iostream>

#include <iomanip>

using namespace std;

class Base

{

private:

typedef struct

{

unsigned code; // шифр датчика

string name; // имя датчика

} sensor;

typedef struct

{

unsigned code; // шифр подсистемы

string name; // имя подсистемы

} subsystem;

typedef struct

{

sensor\* sensor\_idx; // индекс датчика - указатель

unsigned subsys\_idx; // индекс подсистемы - порядковый номер в списке подсистем

} link;

list<sensor\*> list\_sens; // список датчиков

list<subsystem\*> list\_subsys; // список подсистем

list<link\*> list\_links; // список связей между датчиками и подсистемами

// получение индекса датчика по его шифру

sensor\* getSensonrIndex(unsigned sensor\_code)

{

for (sensor\*& sensor : list\_sens)

{ // в цикле по списку датчиков

if (sensor->code == sensor\_code) return sensor; // если найден датчик с нужным шифром - возвращаем его

}

return NULL; // цикл прошел, а датчика так и не нашли, то вернули NULL

}

// получение индекса подсистемы по ее шифру

int getSubsystemIndex(unsigned subsystem\_code)

{

int i = 0;

for (subsystem\*& subsystem : list\_subsys)

{

// в цикле по списку подсистем

if (subsystem->code == subsystem\_code) return i;

// возвращаем не указатель, а его номер в списке

i++;

}

return -1;

}

// получение связки по индексу датчика

link\* getLink(sensor\* sensor)

{

for (link\*& link : list\_links)

{

if (link->sensor\_idx == sensor) return link; // ищем не по шифру, а по "индексу" датчика

}

return NULL;

}

// получение связки по индексу подсистемы

link\* getLink(unsigned subsystem\_idx)

{

for (link\*& link : list\_links)

{

if (link->subsys\_idx == subsystem\_idx) return link; // сравниваем индекс подсистемы с текущим в связке

}

return NULL;

}

public:

~Base()

{

for (sensor\*& sensor : list\_sens) delete sensor; // проходим в цикле по списку датчиков и удаляем каждый

for (subsystem\*& subsystem : list\_subsys) delete subsystem;

for (link\*& link : list\_links) delete link;

}

// статический метод ввода шифра с указанием размерности

static unsigned inputCode(char code\_length) {

string res;

cin >> res;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

char err\_message[256];

sprintf\_s(err\_message, "Вводимое значение должно быть %d-значным целым числом", code\_length);

// сохраняет последовательнось символов и значений

if (res.length() != code\_length || res[0] == '0') throw err\_message;

// если длина шифра не равна нужной, сразу выдаём ошибку

unsigned long tmp;

try

{

tmp = stoul(res); // пробуем получить из введенной строки число

}

catch (invalid\_argument)

{

cout << "Ошибка" << endl;

// throw err\_message; // если не получилось, значит это не число, ошибка

}

return tmp;

}

// добавление датчика

void addSensor(unsigned sensor\_code, string sensor\_name) {

if (getSensonrIndex(sensor\_code)) throw "Датчик с таким шифром уже имеется в базе"; // проверяем, нет ли уже такого

sensor\* sensor\_ptr = new sensor; // выделяем память под новый датчик

sensor\_ptr->code = sensor\_code; // присваиваем ему введенный шифр

sensor\_ptr->name = sensor\_name;

list\_sens.push\_back(sensor\_ptr); // добавляем датчик в список list

}

// добавление подсистемы

void addSubsys(unsigned subsystem\_code, string subsystem\_name) {

if (getSubsystemIndex(subsystem\_code) > -1) throw "Подсистема с таким шифром уже есть в базе";

subsystem\* subsystem\_ptr = new subsystem;

subsystem\_ptr->code = subsystem\_code;

subsystem\_ptr->name = subsystem\_name;

list\_subsys.push\_back(subsystem\_ptr);

}

//привязка датчика к подсистеме

void Bind(unsigned sensor\_code, unsigned subsystem\_code) {

sensor\* sensor\_ptr = getSensonrIndex(sensor\_code); // получаем указатель на датчик по шифру

if (sensor\_ptr == NULL) throw "Нет датчика с таким шифром"; // если вернуло null, то выкидываем исключение

if (getLink(sensor\_ptr)) throw "Датчик уже привязан к подсистеме"; // проверяем, не включен ли датчик уже куда-то

int subsystem\_idx = getSubsystemIndex(subsystem\_code); // получаем индекс подсистемы по шифру

if (subsystem\_idx == -1) throw "Нет подсистемы с таким шифром"; // если вернуло -1, то кидаем исключение - нет такой

link\* link\_ptr = new link; // создаем запись типа link (связь м/у датч. и подсист.)

link\_ptr->sensor\_idx = sensor\_ptr; // записываем в нее указатель на датчик (индекс датчика)

link\_ptr->subsys\_idx = subsystem\_idx; // и указатель подсистемы (индекс подсистемы)

list\_links.push\_back(link\_ptr); // запись добавляем в список связок

}

//отвязка датчика от подсистем

void unBind(unsigned sensor\_code) {

sensor\* sensor\_ptr = getSensonrIndex(sensor\_code); // получаем указатель на датчик по шифру

if (!getLink(sensor\_ptr)) throw "Датчик не привязан ни к одной подсистеме"; // проверяем, включен ли в подсистему

for (list<link\*>::iterator it = list\_links.begin(), end = list\_links.end(); it != end; it++) { // итерируясь в цикле по списку

if ((\*it)->sensor\_idx == sensor\_ptr) { // если у текущей "связки" индекс датчика равен нужному,

list\_links.erase(it); // удаляем связку - теперь датчик отвязан от подсистемы

return; // останавливаемся, чтобы не перебирать зря остальные

}

}

// удаление подсистемы

void deleteSubsys(unsigned subsystem\_code) {

int subsystem\_idx = getSubsystemIndex(subsystem\_code); // получаем индекс подсистемы по шифру

if (subsystem\_idx == -1) throw "Нет подсистемы с таким шифром"; // если не найдена, сообщаем об этом

for (list<link\*>::iterator it = list\_links.begin(), end = list\_links.end(); it != end;) { // итерируясь по списку "связок"

if ((\*it)->subsys\_idx == subsystem\_idx) { // проверяем, если у связки нужный индекс подсистемы

list<link\*>::iterator next\_it = next(it); // получаем ссылку на следующий эл-т списка (итератор)

list\_links.erase(it); // удаляем связку (отвязываем датчик)

it = next\_it; // присваиваем ее переменной цикла (переход к следующему)

}

else {

it++; // иначе - просто переходим к следующему итератору

if ((\*it)->subsys\_idx > subsystem\_idx) { // для инд.записей с большими индексами подсистем

(\*it)->subsys\_idx--; // уменьшаем индекс подсистемы,

} // т.к. удалили одну подсистему - уменьшился номер у следующих

}

//удаляем записи потому что итератор идёт дальше даже после удаления записи, а не останавливается.

}

int i = 0;

for (list<subsystem\*>::iterator it = list\_subsys.begin(), end = list\_subsys.end(); it != end; it++, i++) {

if ((\*it)->code == subsystem\_code) { // так же проходим по списку подсистем, если нашли нужную

list\_subsys.erase(it); // то удаляем ее

break; // и останавливаем цикл

}

}

}

// удаление всех неиспользуемых датчиков

void deleteUnused() {

for (list<sensor\*>::iterator it = list\_sens.begin(), end = list\_sens.end(); it != end;) { //проходим в цикле по датчикам

sensor\* sensor\_ptr = \*it; // получаем указатель на текущий датчик

if (!getLink(sensor\_ptr)) { // если он не привязан ни к какой подсистеме то удаляем

list<sensor\*>::iterator next\_it = next(it); // и получаем ссылку на следующий итератор

list\_sens.erase(it); // удаляем датчик

it = next\_it; // переходим к следующему

}

else it++; // иначе - просто переходим к следующему

}

}

// вывод подсистем, к которым не привязаны датчики

void printSubsysNoSens() {

int num\_output = 0; // счетчик количества выводимых подсистем

unsigned idx = 0; // переменная для индекса (номера) очередной подсистемы

for (subsystem\*& subsys : list\_subsys)

{ // проходим по списку подсистем

if (!getLink(idx))

{ // если не нашли индексную запись(связку) для подсистемы

cout << subsys->name << endl; // то датчиков у нее нет, выводим название

num\_output++; // увеличили счетчик

}

idx++; // увеличили номер, переходим к следующей

}

if (!num\_output)

cout << "Нет таких подсистем" << endl;

}

// вывод списка датчиков

void printSensors()

{

if (list\_sens.begin() != list\_sens.end())

{ // если список датчиков не пуст

cout << "Датчики:" << endl;

cout << left << setw(20) << "Шифр" << setw(40) << "Название" << endl;

for (sensor\*& sensor\_ptr : list\_sens)

{ // в цикле по списку

cout << left << setw(20) << sensor\_ptr->code << setw(40) << sensor\_ptr->name << endl; // выводим датчики

}

cout << endl;

}

}

// вывод списка подсистем

void printSubsys()

{

if (list\_subsys.begin() != list\_subsys.end())

{

cout << "Подсистемы:" << endl;

cout << left << setw(20) << "Шифр" << setw(40) << "Название" << endl;

for (subsystem\*& subsystem\_ptr : list\_subsys)

{

cout << left << setw(20) << subsystem\_ptr->code << setw(40) << subsystem\_ptr->name << endl;

}

cout << endl;

}

}

};

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include <Windows.h>

#include "Base.h"

#include <iostream>

int main() try

{

SetConsoleCP(1251); // Кодировка windows-1251 на ввод

SetConsoleOutputCP(1251); // Кодировка windows-1251 на вывод

setlocale(LC\_ALL, ""); // Правка локали

Base db;

char item;

do {

system("cls");

db.printSensors();

db.printSubsys();

cout << " " << endl;

cout << "0 - выход" << endl;

cout << "1 - добавление подсистемы" << endl;

cout << "2 - добавление датчика" << endl;

cout << "3 - включение датчика в подсистему" << endl;

cout << "4 - исключение датчика из подсистемы" << endl;

cout << "5 - удаление подсистемы" << endl;

cout << "6 - удаление неиспользуемых датчиков" << endl;

cout << "7 - вывод всех подсистем, не содержащих датчики" << endl;

try {

if (!(cin >> item)) {

throw "Пункт меню должен быть задан символом";

cin.clear(); // очищение флагов, чтобы будущие операции ввода-вывода работали правильно

cin.ignore(cin.rdbuf()->in\_avail()); //очистка данных, функция rdbuf возвращет буфер, а in\_avail - количество байт в нем.

}

switch (item) {

case '1':

{

cout << "Введите шифр подсистемы: ";

unsigned cipher = Base::inputCode(2);

cout << "Введите название подсистемы: ";

string name;

getline(cin, name); // ввод данных из потока

db.addSubsys(cipher, name);

break;

}

case '2':

{

cout << "Введите шифр датчика: ";

unsigned cipher = Base::inputCode(3);

cout << "Введите название датчика: ";

string name;

getline(cin, name);

db.addSensor(cipher, name);

break;

}

case '3':

{

cout << "Введите шифр подсистемы: ";

unsigned subsys\_cipher = Base::inputCode(2);

cout << "Введите шифр датчика: ";

unsigned sensor\_cipher = Base::inputCode(3);

db.Bind(sensor\_cipher, subsys\_cipher);

cout << "Подсистема создана" << endl;

break;

}

case '4':

{

cout << "Введите шифр датчика: ";

unsigned sensor\_cipher = Base::inputCode(3);

db.unBind(sensor\_cipher);

cout << "Датчик создан" << endl;

break;

}

case '5':

{

cout << "Введите шифр подсистемы: ";

unsigned subsys\_cipher = Base::inputCode(2);

db.deleteSubsys(subsys\_cipher);

cout << "Подсистема удалена" << endl;

break;

}

case '6': {

db.deleteUnused();

cout << "Неиспользуемые датчики удалены" << endl;

break;

}

case '7': {

cout << "Результат: " << endl;

db.printSubsysNoSens();

break;

}

}

}

catch (const char\* except) {

cout << "Ошибка: " << except << endl;

}

cout << endl;

system("pause");

} while (item != '0');

return 0;

}

catch (...) {

cout << "Ошибка!" << endl;

}

**Пример выполнения программы**

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание