

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

**Отчет
по лабораторной работе № 6
по дисциплине
«Объектно-ориентированное программирование»
на тему
«Исключения, контейнерные классы STL»**

Выполнил:
студент гр. 143
Вербицкая И. С.

Проверил:
Антипов О.В.

Рязань 2023

Задание

В данной лабораторной работе необходимо модифицировать программу из лабораторной работы №4, добавив в неё исключения во всех необходимых местах (**не менее 8 случаев использования**), также заменить все места использования строковых\числовых констант для идентификации типов объектов на **enum class**, и заменить все рукописные контейнерные классы, на класс **std::vector**. Для манипуляции с элементами vector-а использовать исключительно итераторы.

Задание

В данной лабораторной работе необходимо разработать классы в соответствии со своей предметной областью (по вариантам ниже). Все задания состоят из набора сущностей, которые будет необходимо оформить в правильные отношения между объектами. В вариантах заданий описаны только обязательные сущности, можно добавлять свои по необходимости.

В каждом задании есть сущность базового абстрактного класса, которую реализуют классы-наследники. Также есть композитная сущность – это класс, который владеет односвязным списком указателей на базовый абстрактный класс, может добавлять и удалять экземпляры, а также выполнять другие операции над ними (через полиморфный вызов методов). При добавлении\удалении объектов используется динамическое выделение памяти через new\delete. Список необходимо реализовать вручную, через указатели.

В программе необходимо предусмотреть консольный интерфейс, дающий возможность случайной генерации **N** объектов-сущностей из варианта задания и их добавления в композитную сущность, возможность удаления объектов по указанному номеру из списка, а также отображения результатов выполнения действий над объектами.

Вариант 5.

Сущности: деталь, механизм, изделие, узел, устройство, датчик.

Действия: расчёт стоимости и веса всего устройства; расчёт среднего количества деталей на каждое устройство; расчёт общего количества датчиков и их суммарную стоимость по каждому узлу; вывод информации по каждому объекту, включая его состав.

Описание сущностей:

Любое изделие имеет вес и стоимость. Стоимость складывается из стоимости материалов и работы по изготовлению.

Деталь является частью механизма. Механизм состоит из нескольких деталей. Также механизм содержит информацию о расчётом времени износа. Вес и стоимость механизма складывается из соответственно веса и стоимости его деталей. Может включать в себя несколько датчиков.

Каждый датчик повышает стоимость устройства.

Узел состоит из нескольких механизмов. Вес, стоимость, расчётное время износа узла складывается из соответственно веса, стоимости и износа его механизмов. Кроме того, есть коэффициент сложности транспортировки: если вес превышает 20 кг. – стоимость повышается на 10%, если превышает 50 кг. – то на 20%.

Датчик имеет стоимость, вес не учитывается. Имеет тип замеряемой характеристики (температура, вибрация и т.д.)

Все объекты имеют названия.

Описание структуры программы

Выполнение данной лабораторной работы позволило существенно сократить объем кода программы.

Так, например, была убрана композитная сущность односвязного списка и заменена на динамический массив vector из STL. Однако структура программы не изменилась (рисунок 1-2):

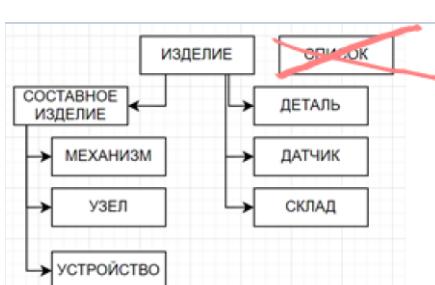


Рисунок 1 – Иерархия наследования

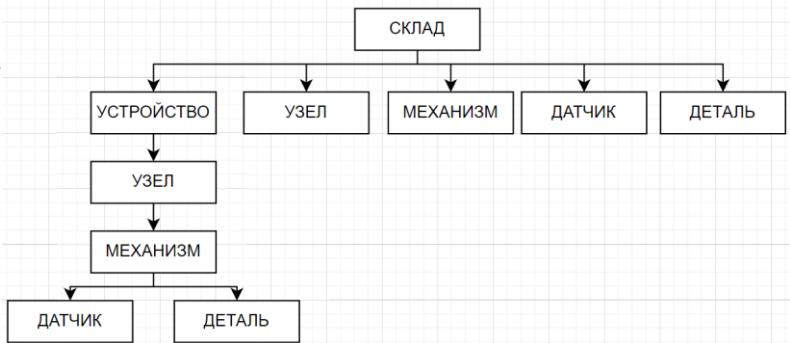


Рисунок 2 – Иерархия хранения сущностей (состав изделий)

Кроме того, использование исключений также сократило количество кода программы. Так, например, выглядела процедура присоединения одного элемента к другому в 4-ой лабораторной работе:

```
17 |     virtual bool AddContent(Item* item) override {  
18 |         if (strcmp(item->getType(),"МЕХАНИЗМ")==0) {  
19 |             Item::AddContent(item);  
20 |             return true;  
21 |         }  
22 |         else return false;  
23 |     }
```

А так – в 6-ой:

```
21 |     virtual void AddContent(Item* item) override {  
22 |         if (item->Type != ItemType::MECHANISM) throw Error("К УЗЛАМ можно присоединять только МЕХАНИЗМЫ");  
23 |         Item::AddContent(item);  
24 |     }
```

Дополнительно был добавлен модуль Error.h с классом Error, экземпляры которого используются в качестве выбрасываемых исключений в программе.

Листинг программы:

Файл Error.h

```
1 //Класс исключений с необходимым текстом
2 class Error
3 {
4     const char* info;
5 public:
6     Error(const char* info) : info(info) {}
7     const char* getInfo() { return info; }
8 };
9
```

Файл Item.h

```
1 #pragma once
2 #include <iostream>
3 #include <strings.h>
4 #include <vector>
5 #include "Error.h"
6
7 #define MAX_STR 256
8 using namespace std;
9
10 //названия типов изделий
11 enum class ItemType {
12     DETAILED, SENSOR, MECHANISM, GROUP, DEVICE, STORAGE
13 };
14 // перегрузка оператора << класса ostream для ItemType
15 // для вывода в стандартном потоке вывода
16 ostream& operator<<(ostream& os, const ItemType& t);
17
18
19 //ИЗДЕЛИЕ - базовый абстрактный класс
20 class Item {
21
22     private:
23
24         char Name[MAX_STR]; //название изделия
25         static int NextId; //следующий присваиваемый серийный номер изделия
26
27     protected:
28
29         int Id; //серийный номер изделия
30         vector<Item*> Content; //массив указателей на элементы, входящие в изделие
31
32     public:
33
34         ItemType Type; //тип изделия
35         Item(const char* name);
36         virtual ~Item();
37
38         //ПОИСК, УДАЛЕНИЕ, ДОБАВЛЕНИЕ
39         virtual void AddContent(Item* item)=0; //добавление к Content нового элемента
40         void DelContent(int const &N); //удаляет из вложенного состава Content изделия по его номеру
41         void DelTopContent(int const &N); //удаляет из непосредственного состава изделия (без состава состава) изд по номеру
42         Item* SearchContent(int const &N) const; //возвр указатель на элемент, если найдет его во вложенном составе изделия
43         void Clear(); //удаляет всё вложенное содержимое изделия
44
45         //ФУНКЦИИ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЙ ДЛЯ ИЗДЕЛИЯ
46         virtual int Weight() const = 0; //общая масса изделия
47         virtual float Price() const = 0; //стоимость изделия
48         virtual int Time() const = 0; //расчетное время износа изделия
49
50         // ПРОЦЕДУРЫ ВЫВОДА ДАННЫХ
51         virtual void printAbout() const = 0; //вывод полной информации об изделии
52         void printSimpleAbout() const; //вывод номера, типа и названия изделия
53         void printContent() const; //вывод информации об изделиях в составе данного
54         void printSimpleContent() const; //вывод частичной информации об изделиях в составе данного
55         void printSomeContent(int const &N) const; //вывод инф-ии об изделии с номером N в составе данного
56
57         //ФУНКЦИИ ПРОВЕРКИ, ЕСТЬ ЛИ ЭЛЕМЕНТ В СОСТАВЕ ДРУГОГО
58         bool InTopContent(int const &N) const; //определяет, входит ли изделие в состав данного непосредственно
59         bool InContent(int const &N) const; //опр, входит ли изделие в состав данного через состав его состава
60     };
61
```

Файл Item.cpp

```
1 #include "Item.h"
2
3     ostream& operator<<(ostream& os, const ItemType& t) {
4         switch (t) {
5             case ItemType::DETAIL: { return os << "ДЕТАЛЬ"; break; }
6             case ItemType::SENSOR: { return os << "ДАТЧИК"; break; }
7             case ItemType::MECHANISM: { return os << "МЕХАНИЗМ"; break; }
8             case ItemType::GROUP: { return os << "УЗЕЛ"; break; }
9             case ItemType::DEVICE: { return os << "УСТРОЙСТВО"; break; }
10        }
11    };
12
13 //инициализация static-переменной NextId
14 int Item::NextId = 0;
15
16     Item::Item(const char* name) {
17         strcpy(Name, name);
18         Id = NextId;
19         NextId++;
20     }
21
22     Item::~Item() {
23         Clear();
24     }
25
26     void Item::Clear() {
27         // vector<Item*>::iterator it;
28         auto it = Content.begin();
29         while (it != Content.end()) {
30             (*it)->Clear();
31             ++it;
32         }
33         Content.clear();
34     }
35
36     void Item::AddContent(Item* item) {
37         Content.push_back(item);
38     }
39
40     void Item::DelContent(int const &N) {
41         auto it = Content.begin();
42         while (it != Content.end()) {
43             if ((*it)->Id == N) {
44                 (*it)->Clear(); //удаляем содержимое
45                 Content.erase(it); //удаляем сам узел
46                 break;
47             }
48             else (*it)->DelContent(N);
49             ++it;
50         }
51     }
52 }
```

```
53     void Item::DelTopContent(int const &N) {
54         auto it = Content.begin();
55         while (it != Content.end()) {
56             if ((*it)->Id == N) {
57                 (*it)->Clear(); //удаляем содержимое
58                 Content.erase(it); //удаляем само изделие
59                 break;
60             }
61             ++it;
62         }
63     }
64
65     Item* Item::SearchContent(int const &N) const {
66         Item* R = nullptr;
67         vector<Item*>::const_iterator it;
68         it = Content.begin();
69         if (!Content.empty()) {
70             while (it != Content.end()) {
71                 if ((*it)->Id == N) {
72                     R = *it;
73                 }
74                 else {
75                     R = (*it)->SearchContent(N);
76                 }
77                 if (R!=nullptr) return R;
78                 ++it;
79             }
80         }
81         return R;
82     }
83
84     void Item::printAbout() const {
85         if (!Content.empty()) {
86             cout << " _____" << endl;
87             cout << "/ \\" << endl;
88             printContent();
89             cout << "\n\\____/\\" << endl;
90         }
91         cout << "\nИЗДЕЛИЕ № " << Id << ":" << Type << endl;
92         cout << "| Название: " << Name << endl;
93         cout << "| Стоимость: " << Price() << " руб" << endl;
94         if (Weight() > 0)
95             cout << "| Вес: " << Weight() << " г" << endl;
96     }
97
98     void Item::printContent() const {
99         vector<Item*>::const_iterator it;
100        it = Content.begin();
101        while (it != Content.end()) {
102            (*it)->printAbout();
103            ++it;
104        }
105    }
```

```
104     }
105   }
106 
107   void Item::printSimpleAbout() const {
108     cout << Type << " - \\" << Name << "\" - № " << Id << endl;
109   }
110 
111   void Item::printSimpleContent() const {
112     vector<Item*>::const_iterator it;
113     it = Content.begin();
114     while (it != Content.end()) {
115       (*it)->printSimpleAbout();
116       ++it;
117     }
118   }
119 
120   void Item::printSomeContent(int const &N) const {
121     vector<Item*>::const_iterator it;
122     it = Content.begin();
123     while (it != Content.end()) {
124       if ((*it)->Id == N) (*it)->printAbout();
125       else (*it)->printSomeContent(N);
126       ++it;
127     }
128   }
129 
130   bool Item::InTopContent(int const &N) const {
131     bool F = false;
132     vector<Item*>::const_iterator it;
133     it = Content.begin();
134     while (it != Content.end()) {
135       if ((*it)->Id == N) F = true;
136       ++it;
137     }
138     return F;
139   }
140 
141   bool Item::InContent(int const &N) const {
142     bool F = false;
143     vector<Item*>::const_iterator it;
144     it = Content.begin();
145     while (it != Content.end()) {
146       if ((*it)->Id == N) F = true;
147       else F = (*it)->InContent(N);
148       if (F) return F;
149       ++it;
150     }
151   }
152 }
```

Файл Detail.h

```
1 #pragma once
2 #include "Item.h"
3
4 class Detail: public Item { //ДЕТАЛЬ
5     int Mass; //масса детали
6     int Cost; //стоимость детали
7
8 public:
9
10    Detail(const char* name, const int weight, const int money): Item(name) {
11        if ((weight <= 0) || (money <= 0)) throw Error("Недопустимые значения параметров!");
12        Mass=weight;
13        Cost=money;
14        Type=ItemType::DETAIL;
15    }
16    ~Detail() {}
17
18    virtual void printAbout() const override { Item::printAbout(); }
19    virtual void AddContent(Item* item) override { throw Error("К ДЕТАЛЯМ ничего нельзя присоединять"); }
20    virtual int Weight() const override { return Mass; }
21    virtual float Price() const override { return Cost; }
22    virtual int Time() const override { return 0; }
23
24};
25
26
27
```

Файл Sensor.h

```
1 #pragma once
2 #include "Item.h"
3
4 using namespace std;
5
6 class Sensor: public Item { //ДАТЧИК
7     float Cost; //стоимость датчика
8     char Measure[MAX_STR]; //тип замеряемой характеристики
9
10 public:
11
12    Sensor(const char* name,const char* measure,const int money): Item(name) {
13        if (money <= 0) throw Error("Недопустимые значения параметров!");
14        strcpy(Measure, measure);
15        Cost=money;
16        Type=ItemType::SENSOR;
17    }
18    ~Sensor() {}
19
20    virtual void AddContent(Item* item) override { throw Error("К ДАТЧИКАМ ничего нельзя присоединять"); }
21    virtual int Weight() const override { return 0; }
22    virtual float Price() const override { return Cost; }
23    virtual int Time() const override { return 0; }
24    virtual void printAbout() const override {
25        Item::printAbout();
26        cout << "| Характ.: " << Measure << endl;
27    }
28
29};
```

Файл Storage.h

```
1 #pragma once
2 #include "Item.h"
3 #include "Random.h"
4
5 class Storage: public Item { //СКЛАД
6
7     public:
8
9     Storage(const char* name): Item(name) { Type = ItemType::STORAGE; }
10    ~Storage() { /*Content.Clear(); */ }
11
12    virtual void printAbout() const final;
13    virtual int Weight() const override {}
14    virtual float Price() const override {}
15    virtual int Time() const override {}
16    virtual void AddContent(Item* item) override { Item::AddContent(item); }
17
18    void printAboutContent(int const &Key) const; //вывод конкретного изделия на складе
19    void printSimpleStorage() const; //вывод изделий на складе в упрощенном виде
20    void DelSomeContent(int const &Key); //удаление конкретного изделия на складе
21    void DelAllContent(); //полное удаление элементов склада
22    void Joining (int const &What, int const &ToWhich); //присоединение элемента What в состав ToWhich
23    void Gen (int const &N); //добавление на склад N сгенерированных сущностей
24    void GenItem(); //добавление на склад сгенерированной сущности
25};
```

Файл Storage.cpp

```
1 #include "Storage.h"
2
3 using namespace std;
4
5 void Storage::printAbout() const {
6     if (Content.empty()) throw Error("На складе пока нет изделий!");
7     cout << "\n----- СКЛАД ----- \n" << endl;
8     printContent();
9 }
10
11 void Storage::printAboutContent(int const &Key) const {
12     if (Content.empty()) throw Error("На складе пока нет изделий!");
13     if (!(InContent(Key))) throw Error("Такого изделия нет!");
14     cout << "\n---- ИЗДЕЛИЕ № " << Key << " ---- \n" << endl;
15     printSomeContent(Key);
16 }
17
18 void Storage::printSimpleStorage() const {
19     if (Content.empty()) throw Error("На складе пока нет изделий!");
20     cout << "\n----- СКЛАД ----- \n" << endl;
21     printSimpleContent();
22 }
23
24 void Storage::DelSomeContent(int const &Key) {
25     if (Content.empty()) throw Error("На складе пока нет изделий!");
26     if (!(InContent(Key))) throw Error("Такого изделия нет!");
27     DelContent(Key);
28     cout << "Удаление прошло успешно!" << endl;
29 }
30
31 void Storage::DelAllContent() {
32     if (Content.empty()) throw Error("Склад пуст!");
33     Clear();
34     cout << "Склад успешно уничтожен!" << endl;
35 }
```

```

36 void Storage::Joining (int const &what, int const &toWhich) { // что - куда
37     if (Content.empty()) throw Error("На складе пока нет изделий!");
38     if (!InContent(what)) throw Error("На складе нет элемента, который вы хотите присоединить!");
39     if (!InTopContent(what)) throw Error("Присоединять можно только изделия, лежащие на складе,\nт.е. те, которые еще не присоединены к другим!");
40     if (!InContent(toWhich)) throw Error("На складе нет элемента, в состав которого вы хотите включить изделие!");
41     try {
42         SearchContent(toWhich)->AddContent(SearchContent(what));
43         DelTopContent(what);
44         cout << "\nПрисоединение прошло успешно!\n" << endl;
45     }
46     catch (Error& E) { cout << E.getInfo() << endl; }
47     catch(...) { cout << "\nПроизошла неизвестная ошибка!\n" << endl; }
48 }
49
50 void Storage::Gen(int const &N) { for (int i=0;i<N;i++) GenItem(); }
51
52 void Storage::GenItem() {
53     int K = 1+rand()%5;
54     switch (K) {
55         case 1: {AddContent(RandomDetail()); break;}
56         case 2: {AddContent(RandomSensor()); break;}
57         case 3: {AddContent(RandomMechanism()); break;}
58         case 4: {AddContent(RandomGroup()); break;}
59         case 5: {AddContent(RandomDevice()); break;}
60     }
61 }
62 }
63

```

Файл ComplexItem.h

```

1 #pragma once
2 #include "Item.h"
3
4 //предназначен для наследования сущностям, которые могут состоять из других
5 class ComplexItem: public Item { //СОСТАВНОЕ ИЗДЕЛИЕ
6
7 protected:
8
9     //сумму масс изделий, входящих в состав изделия
10    int ContentWeight() const;
11    //сумма стоимости всех изделий, входящих в состав данного
12    int ContentCost() const;
13    //расчётоное время износа изделия исходя из его состава
14    int ContentTime() const;
15
16 public:
17
18     ComplexItem(const char* name): Item(name) {}
19     virtual ~ComplexItem() { /*Content.Clear(); */ }
20
21     virtual int Weight() const final { return ContentWeight(); }
22     virtual void printAbout() const final {
23         Item::printAbout();
24         cout << "| Время износа: " << Time() << " дней" << endl;
25     }
26 };
27

```

Файл ComplexItem.cpp

```

1 #include "ComplexItem.h"
2
3 int ComplexItem::ContentWeight() const {
4     int S=0;
5     vector<Item*>::const_iterator it;
6     it = Content.begin();
7     while (it != Content.end()) {
8         S += (*it)->Weight();
9         ++it;
10    }
11    return S;
12 }
13
14 int ComplexItem::ContentCost() const {
15     int S=0;
16     vector<Item*>::const_iterator it;
17     it = Content.begin();
18     while (it != Content.end()) {
19         S += (*it)->Price();
20         ++it;
21     }
22     return S;
23 }
24
25 int ComplexItem::ContentTime() const {
26     int max=0;
27     vector<Item*>::const_iterator it;
28     it = Content.begin();
29     while (it != Content.end()) {
30         if ((*it)->Time() > max) max=(*it)->Time();
31         ++it;
32     }
33     return max;
34 }

```

Файл Mechanism.h

```
1 #pragma once
2 #include "ComplexItem.h"
3
4 class Mechanism: public ComplexItem { //МЕХАНИЗМ
5
6     private:
7
8         int Timer; //Время износа датчика
9
10    public:
11
12    Mechanism(const char* name,const int time): ComplexItem(name) {
13        if (time <= 0) throw Error("Недопустимые значения параметров!");
14        Timer=time;
15        Type=ItemType::MECHANISM;
16    }
17    ~Mechanism() {}
18
19    virtual float Price() const override { return ContentCost(); }
20    virtual int Time() const override { return Timer; }
21    virtual void AddContent(Item* item) override {
22        if (item->Type != ItemType::SENSOR && item->Type != ItemType::DETAIL)
23            throw Error("К МЕХАНИЗМАМ можно присоединять только ДАТЧИКИ и ДЕТАЛИ");
24        Item::AddContent(item);
25    }
26}
27
28
```

Файл Group.h

```
1 #pragma once
2 #include "ComplexItem.h"
3
4 class Group: public ComplexItem { //УЗЕЛ
5
6     public:
7
8     Group(const char* name): ComplexItem(name) {
9         Type = ItemType::GROUP;
10    }
11    ~Group() {}
12
13    virtual float Price() const override {
14        int S=ContentCost();
15        if (Weight()<=20) {}
16        else if (Weight()<=50) S=S*1.1;
17        else S = S*1.2;
18        return S;
19    }
20    virtual int Time() const override { return ContentTime(); }
21    virtual void AddContent(Item* item) override {
22        if (item->Type != ItemType::MECHANISM) throw Error("К УЗЛАМ можно присоединять только МЕХАНИЗМЫ");
23        Item::AddContent(item);
24    }
25}
```

Файл Device.h

```
1 #pragma once
2 #include "ComplexItem.h"
3
4 class Device: public ComplexItem { //УСТРОЙСТВО
5
6     private:
7     protected:
8     public:
9
10    Device(const char* name): ComplexItem(name) {
11        Type = ItemType::DEVICE;
12    }
13    ~Device() {}
14
15    virtual float Price() const override { return ContentCost(); }
16    virtual int Time() const override { return ContentTime(); }
17    virtual void AddContent(Item* item) override {
18        if (item->Type != ItemType::GROUP) throw Error("К УСТРОЙСТВАМ можно присоединять только УЗЛЫ");
19        Item::AddContent(item);
20    }
21
```

Файл Menu.h

```
1 #pragma once
2 #include "Detail.h"
3 #include "Sensor.h"
4 #include "Mechanism.h"
5 #include "Group.h"
6 #include "Device.h"
7 #include "Storage.h"
8
9 void MainMenu(Storage* Store); //главное меню
10 void OutputMenu(Storage* const Store); //меню вывода (вывод сущностей)
11 void DeletionMenu(Storage* Store); //меню удаления (удаление изделий)
12 void AdditionMenu(Storage* Store); //меню добавления (внесение новых сущностей вручную)
13 void MovementMenu(Storage* Store); //меню перемещения (добавление одних элементов в состав других)
14 void GenerationMenu(Storage* Store); //меню генерации (случайных сущностей на склад)
15
```

Файл Menu.cpp

```
1 #include "Menu.h"
2 using namespace std;
3
4 void MainMenu(Storage* Store) {
5     int Key; //номер вводимой команды
6     while (true) {
7         system("cls");
8         cout << "\n----- ДОБРО ПОЖАЛОВАТЬ В СИМУЛЯТОР СБОРЩИКА НА ЗАВОДЕ! : ) ----- \n" << endl;
9         cout << "Введите номер команды" << endl;
10        cout << "1 - Посмотреть содержимое склада (вывод данных)" << endl;
11        cout << "2 - Закупить случайные изделия (генерация данных)" << endl;
12        cout << "3 - Уничтожить изделие (удаление данных)" << endl;
13        cout << "4 - Купить новое изделие (внесение новых данных)" << endl;
14        cout << "5 - Собрать изделие (присоединить изделие со склада в состав другого изделия)" << endl;
15        cout << "6 - Закончить рабочий день (завершение работы программы)\n" << endl;
16        cin >> Key;
17        switch (Key) {
18            case 1: {
19                OutputMenu(Store);
20                break;
21            }
22            case 2: {
23                GenerationMenu(Store);
24                break;
25            }
26            case 3: {
27                DeletionMenu(Store);
28                break;
29            }
30            case 4: {
31                AdditionMenu(Store);
32                break;
33            }
34            case 5: {
35                MovementMenu(Store);
36                break;
37            }
38            case 6: {
39                try {
40                    Store->DelAllContent();
41                    cout << "Потому что существует лишь в динамической памяти, а не на самом деле :)" << endl;
42                }
43                catch (Error& E) { }
44                catch (...) { cout << "Произошла неизвестная ошибка!" << endl; }
45                cout << "\nХочется верить, что Вам понравилось это приложение" << endl;
46                cout << "\n(Ведь я в каком-то смысле вложила в него душу)" << endl;
47                cout << "\n((Хотя это сообщение вряд ли кто-то прочитает или заметит))" << endl;
48                exit(0);
49                break;
50            }
51            default:
52                puts("Такой команды не предусмотрено!");
53                system("pause");
54                break;
55        }
56    }
57}
```

```
58 void OutputMenu(Storage* const Store) {
59     system("cls");
60     int Key;
61     cout << "\n----- ВЫВОД ДАННЫХ -----\" << endl;
62     cout << "Что вы хотите посмотреть?" << endl;
63     cout << "1 - Полное содержимое склада (вывод как в команде 2, но еще и с составом)" << endl;
64     cout << "2 - Содержимое склада (вывод номеров и названий изделий, которые еще не присоединены к другим)" << endl;
65     cout << "3 - Сборку конкретного изделия (вывод состава и характеристик)\n" << endl;
66     cin >> Key;
67     try {
68         switch (Key) {
69             case 1: {
70                 Store->printAbout();
71                 break;
72             }
73             case 2: {
74                 Store->printSimpleStorage();
75                 break;
76             }
77             case 3: {
78                 cout << "Изделие с каким номером вы хотите вывести?" << endl;
79                 cin >> Key;
80                 Store->printAboutContent(Key);
81                 break;
82             }
83             default:
84                 puts("Такой команды не предусмотрено!");
85                 break;
86         }
87     }
88     catch (Error& E) { cout << E.getInfo() << endl; }
89     catch(...) { cout << "\nПроизошла неизвестная ошибка!\n" << endl; }
90     cout << "" << endl;
91     system("pause");
92 }
93
94 void DeletionMenu(Storage* Store) {
95     system("cls");
96     int Key;
97     cout << "\n----- УДАЛЕНИЕ ДАННЫХ -----\" << endl;
98     cout << "Что вы хотите уничтожить?" << endl;
99     cout << "1 - Конкретное изделие" << endl;
100    cout << "2 - Сжечь весь склад (удаление всего содержимого)\n" << endl;
101    cin >> Key;
102    try {
103        switch (Key) {
104            case 1: {
105                cout << "Изделие с каким номером вы хотите уничтожить?" << endl;
106                cin >> Key;
107                Store->DelSomeContent(Key);
108                break;
109            }
110            case 2: {
111                Store->DelAllContent();
112                break;
113            }
114        }
115    }
```

```
115     default:
116         puts("Такой команды не предусмотрено!");
117         break;
118     }
119 }
120 catch (Error& E) { cout << E.getInfo() << endl; }
121 catch(...) { cout << "\nПроизошла неизвестная ошибка!\n" << endl; }
122 cout << "" << endl;
123 system("pause");
124 }
125
126 void AdditionMenu(Storage* Store) {
127     int Key;
128     int N1;
129     int N2;
130     char S1[MAX_STR];
131     char S2[MAX_STR];
132
133     system("cls");
134     cout << "\n----- ДОБАВЛЕНИЕ ДАННЫХ -----\" << endl;
135     cout << "Изделие какого типа вы хотите добавить на склад?" << endl;
136     cout << "1 - деталь" << endl;
137     cout << "2 - датчик" << endl;
138     cout << "3 - Механизм" << endl;
139     cout << "4 - Узел" << endl;
140     cout << "5 - Устройство\n" << endl;
141     cin >> Key;
142     try {
143         switch (Key) {
144             case 1: {
145                 cout << "Название детали: ";
146                 cin >> S1;
147                 cout << "Вес: ";
148                 cin >> N1;
149                 cout << "Стоимость: ";
150                 cin >> N2;
151                 Detail* D = new Detail(S1,N1,N2);
152                 Store->AddContent(D);
153                 break;
154             }
155             case 2: {
156                 cout << "Название датчика: ";
157                 cin >> S1;
158                 cout << "Тип замеряемой характеристики: ";
159                 cin >> S2;
160                 cout << "Стоимость: ";
161                 cin >> N1;
162                 Sensor* S = new Sensor(S1,S2,N1);
163                 Store->AddContent(S);
164                 break;
165             }
166             case 3: {
167                 cout << "Название механизма: ";
168                 cin >> S1;
169                 cout << "Время износа: ";
170                 cin >> N1;
171                 Mechanism* M = new Mechanism(S1,N1);
```

```

172     Store->AddContent(M);
173     break;
174 }
175 } case 4: {
176     cout << "Название узла: ";
177     cin >> S1;
178     Group* G = new Group(S1);
179     Store->AddContent(G);
180     break;
181 }
182 } case 5: {
183     cout << "Название устройства: ";
184     cin >> S1;
185     Device* Dev = new Device(S1);
186     Store->AddContent(Dev);
187     break;
188 }
189 default:
190     puts("Такой команды не предусмотрено!");
191     break;
192 }
193 if (!(Key<0 || Key>5)) cout << "Изделие успешно добавлено и ждёт вас на складе!" << endl;
194 }
195 catch (Error& E) { cout << E.getInfo() << endl; }
196 catch(...) { cout << "\nПроизошла неизвестная ошибка!\n" << endl; }
197 cout << "" << endl;
198 system("pause");
199 }
200
201 void MovementMenu(Storage* Store) {
202     system("cls");
203     int Key1, Key2;
204     cout << "\n----- СБОРКА ИЗДЕЛИЙ -----\" << endl;
205     cout << "На складе лежат изделия, которые пока никуда не присоединены." << endl;
206     cout << "Вы можете присоединить одно из них к другому изделию, однако" << endl;
207     cout << "важно соблюдать ПРАВИЛА иерархии:" << endl;
208     cout << "| ДЕТАЛИ и ДАТЧИКИ присоединяются к МЕХАНИЗМАМ" << endl;
209     cout << "| МЕХАНИЗМЫ присоединяются к УЗЛАМ" << endl;
210     cout << "| УЗЛЫ присоединяются к УСТРОЙСТВАМ" << endl;
211     cout << "| УСТРОЙСТВА никуда нельзя присоединить\n" << endl;
212     cout << "\nСейчас на складе лежат следующие изделия, доступные для сборки:" << endl;
213     Store->printSimpleContent();
214     cout << "\nКАКОЕ изделие вы собираетесь присоединять?" << endl;
215     cin >> Key1;
216     cout << "В КАКОЕ изделие вы собираетесь присоединять" << endl;
217     cin >> Key2;
218     try { Store->Joining(Key1,Key2); }
219     catch (Error& E) { cout << E.getInfo() << endl; }
220     catch(...) { cout << "\nПроизошла неизвестная ошибка!\n" << endl; }
221     cout << "" << endl;
222     system("pause");
223 }
224
225 void GenerationMenu(Storage* Store) {
226     system("cls");
227     int Key=0;
228     cout << "\n----- ГЕНЕРАЦИЯ ИЗДЕЛИЙ -----\" << endl;
229     while (Key<=0 || Key>20) {
230         cout << "Сколько изделий необходимо сгенерировать?" << endl;
231         cin >> Key;
232         if (Key<=0) cout << "Введите положительное число!" << endl;
233         if (Key>20) cout << "Лучше не генерировать больше 20 изделий за раз!" << endl;
234     }
235     Store->Gen(Key);
236     cout << "Новые случайные изделия ждут вас на складе!" << endl;
237     cout << "" << endl;
238     system("pause");
239 }
240

```

Файл Random.h

```
1 #pragma once
2 #include "Detail.h"
3 #include "Sensor.h"
4 #include "Mechanism.h"
5 #include "Group.h"
6 #include "Device.h"
7
8 Item* RandomDetail(); //возвращает случайную деталь
9
10 Item* RandomSensor(); //возвр случ датчик
11
12 Item* RandomMechanism(); //возвр сл механизм
13
14 Item* RandomGroup(); //узел
15
16 Item* RandomDevice(); //устройство
```

Файл Random.cpp

```
1 #include "Random.h"
2
3 const char* RandomName[50] = {"m98dAlZW", "aMEtpG4E", "BtTKM5yd", "WQQ2SGnY", "1tnl0o77",
4 "lrbcHjy", "PymAymGq", "du4w9Y18", "tQwx6qwS", "cytuUTiS",
5 "g9ftLEba", "fRJgxPIL", "1JV4zGcW", "1LADne4A", "bS383SgX",
6 "Pr1CoVxe", "pyf5nyt1", "15noS6fv", "5GDFk4CV", "9pSDvoTw",
7 "0xf9pY55", "CFGjEPyT", "etYjDorQ", "fFVBii8", "83rhnjGm",
8 "oDmn7ACy", "Lc3qNL7N", "TMdDsgLi", "dZhpQiOY", "OffiTQhn",
9 "IUYjd34h", "hH21i1u3", "po0232ji", "NTsnhhSU", "M3IKLEh6",
10 "kPzZFtoZ", "la5V8N1u", "2qzDWY4F", "0X8POht6", "20Byw2rR",
11 "jRVddukz", "Gsgg6riZ", "6uFPxnZS", "FCg9buJn", "9Y7a7PFI",
12 "M6AfYDj4", "wbzFeMcR", "xWvWlkWKX", "75xzffjj1", "s9IlPofv"};
13 const char* RandomMeasure[24] = {"Температура", "Давление", "Громкость", "Длина",
14 "Скорость", "Напряжение", "Мощность", "Диаметр",
15 "Вязкость", "Цвет", "Радиоактивность", "Вибрация",
16 "Сила тока", "Напряженность", "Влажность", "Яркость",
17 "Уровень", "Запах", "Индуктивность", "Ёмкость",
18 "Сопротивление", "Вес", "Время", "Загрязнённость"};
19
20 Item* RandomDetail() {
21     char N[MAX_STR];
22     int m,s;
23     strcpy(N,RandomName[rand()%50]);
24     m = 1+rand()%(1000-1+1);
25     s = 100+rand()%(5000-100+1);
26
27     Detail* R = new Detail(N,m,s);
28     return R;
29 }
30
31 Item* RandomSensor() {
32     char N[MAX_STR];
33     char M[MAX_STR];
34     int s;
35     strcpy(N,RandomName[rand()%50]);
36     strcpy(M,RandomMeasure[rand()%24]);
37     s = 100+rand()%(5000-100+1);
38
39     Sensor* R = new Sensor(N,M,s);
40     return R;
41 }
```

```

42
43 ┌ Item* RandomMechanism() {
44     char N[MAX_STR];
45     int t;
46     strcpy(N,RandomName[rand()%50]);
47     t = 30+rand()%(2000-30+1);
48
49     int i;
50     Mechanism* R = new Mechanism(N,t);
51     int n=0+rand()%(3);
52     for (i=0;i<n;i++) R->AddContent(RandomDetail());
53     n=0+rand()%(3);
54     for (i=0;i<n;i++) R->AddContent(RandomSensor());
55     return R;
56 }
57
58 ┌ Item* RandomGroup() {
59     char N[MAX_STR];
60     strcpy(N,RandomName[rand()%50]);
61
62     Group* R = new Group(N);
63     int n=0+rand()%(3);
64     for (int i=0;i<n;i++) R->AddContent(RandomMechanism());
65     return R;
66 }
67
68 ┌ Item* RandomDevice() {
69     char N[MAX_STR];
70     strcpy(N,RandomName[rand()%50]);
71
72     Device* R = new Device(N);
73     int n=0+rand()%(3);
74     for (int i=0;i<n;i++) R->AddContent(RandomGroup());
75     return R;
76 }

```

Файл main.cpp

```

1 #include "Menu.h"
2 #include "Storage.h"
3
4 using namespace std;
5
6 int main() {
7     system("chcp 1251");
8     Storage MyStorage("Склад1");
9     MainMenu(&MyStorage);
10    return 0;
11 }
12

```

Результаты работы программы:

В рамках выполнения данной лабораторной работы №4 было разработано и консольное приложение, позволяющее управлять сущностями из варианта задания и выполнять с ними соответствующие операции (рисунок 3-8). В лабораторной работе №6 это приложение было усовершенствовано, но на уровне пользовательского интерфейса никаких изменений нет – это изменения на программном уровне.

```
----- ДОБРО ПОЖАЛОВАТЬ В СИМУЛЯТОР СБОРЩИКА НА ЗАВОДЕ! :) -----  
Введите номер команды  
1 - Посмотреть содержимое склада (вывод данных)  
2 - Закупить случайные изделия (генерация данных)  
3 - Уничтожить изделие (удаление данных)  
4 - Купить новое изделие (внесение новых данных)  
5 - Собрать изделие (присоединить изделие со склада в состав другого изделия)  
6 - Закончить рабочий день (завершение работы программы)  
  
7  
Такой команды не предусмотрено!  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3 – пример работы с главным меню

```
----- СБОРКА ИЗДЕЛИЙ -----  
На складе лежат изделия, которые пока никуда не присоединены.  
Вы можете присоединить одно из них к другому изделию, однако  
важно соблюдать ПРАВИЛА иерархии:  
| ДЕТАЛИ и ДАТЧИКИ присоединяются к МЕХАНИЗМАМ  
| МЕХАНИЗМЫ присоединяются к УЗЛАМ  
| УЗЛЫ присоединяются к УСТРОЙСТВАМ  
| УСТРОЙСТВА никуда нельзя присоединить  
  
КАКОЕ изделие вы собираетесь присоединять?  
17  
В КАКОЕ изделие вы собираетесь присоединять  
19  
Присоединение прошло успешно!  
  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 - пример работы с меню сборки изделий

```
----- УДАЛЕНИЕ ДАННЫХ -----  
Что вы хотите уничтожить?  
1 - Конкретное изделие  
2 - Сжечь весь склад (удаление всего содержимого)  
  
2  
  
Склад успешно уничтожен!  
  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5 – пример работы с меню удаления

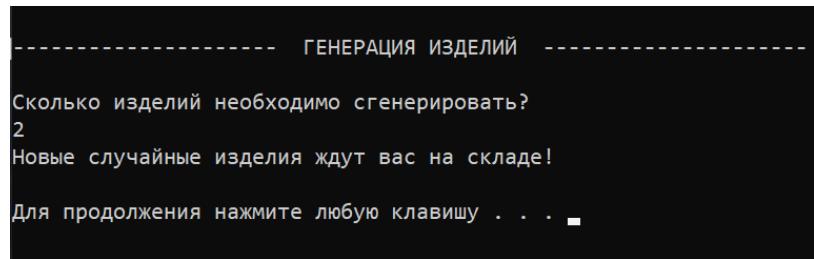


Рисунок 6 – пример работы с меню генерации изделий

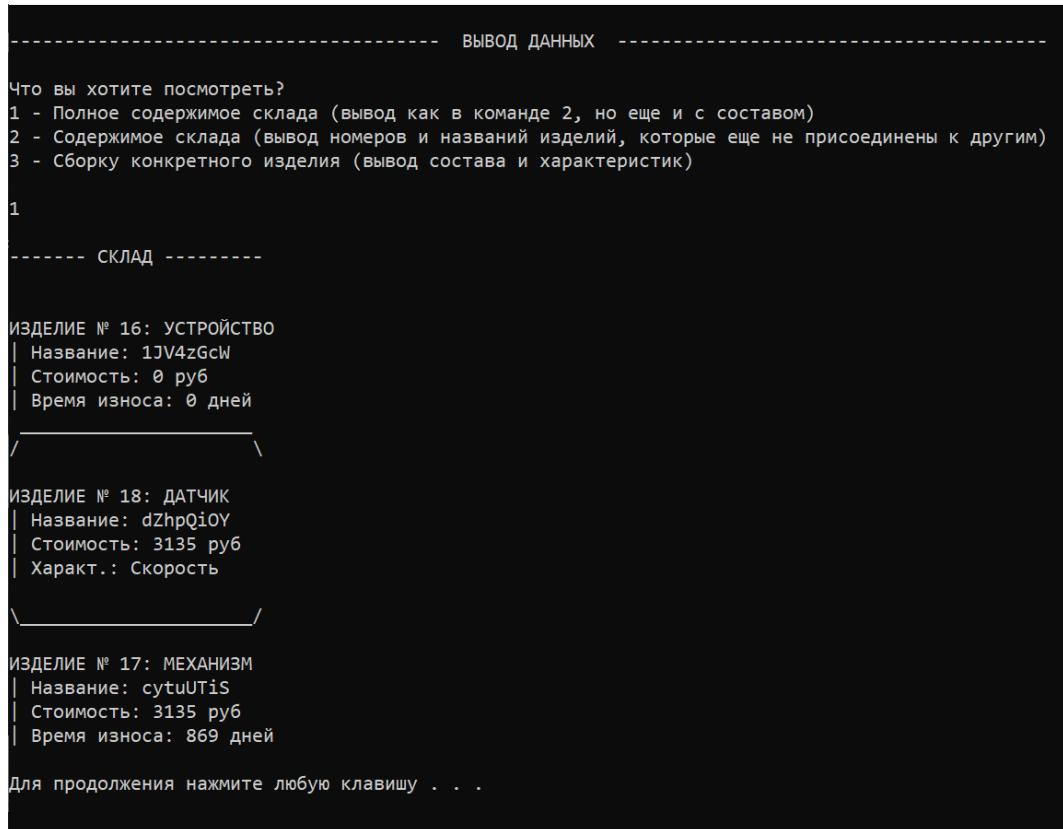


Рисунок 7 – пример работы с меню вывода данных

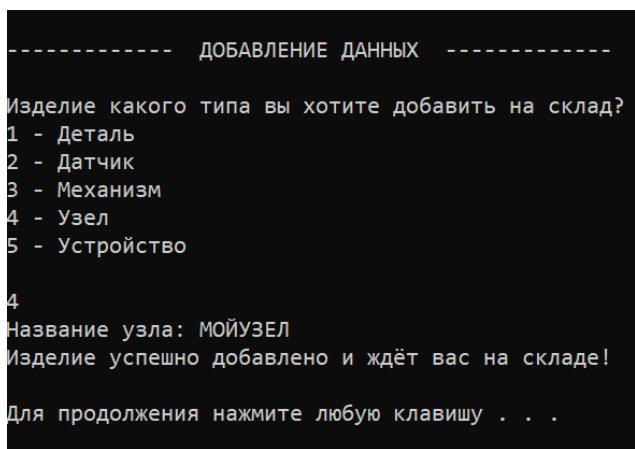


Рисунок 8 – пример работы с меню добавления данных

Приложение справляется с поставленной задачей, работает корректно, без ошибок и не нарушает логику поставленных по варианту условий.