

Лабораторна робота 14. СОРТУВАННЯ

Тема: STL. Алгоритми зміни послідовності. Сорткування. Функтори.

Мета: на практиці порівняти STL-алгоритми, що модифікують послідовність; отримати навички роботи з STL-функторами.

1. Завдання до роботи

Загальне завдання. Поширити попередню лабораторну роботу, додаючи такі можливості діалогового меню:

- об'єднання двох STL-контейнерів типу vector;
- сортувати заданий контейнер з використанням функтора.

Додаткове завдання на оцінку «відмінно»:

Додати можливість об'єднання двох STL-контейнерів типу map. При цьому, якщо в обох контейнерах існують однакові ключі, то значення повинні конкатенуватися, наприклад, якщо є дві мапи для країн:

- Мапа1:
 - Україна : Харків, Київ;
 - Росія: Москва, Белгород;
 - Білорусь: Мінськ, Бобруйськ.

- Мапа2:
 - Польща: Варшава;
 - Росія: Санкт-Петербург;
 - Україна: Харків, Запоріжжя;

то об'єднана мапа повинна містити таке:

- - Україна: Харків, Київ, Запоріжжя;
 - Росія: Москва, Белгород, Санкт-Петербург;
 - Білорусь: Мінськ, Бобруйськ;
 - Польща: Варшава.

2.1. Опис класів

Базовий клас: [CProgram](#).

Клас-спадкоємець: [CMalware](#).

Клас-функтор: [Functor](#).

2.2. Опис змінних

`int` timeOfWork – час роботи програми (змінна класу `CProgram`).
`int` size – розмір програми (змінна класу `CProgram`).
`int` amountOfLines – кількість рядків коду програми (змінна класу `CProgram`).
`int` index – номер програми (змінна класу `CProgram`).
`bool` useInternet – потребує програма Інтернет чи ні (змінна класу `CProgram`).
`string` name – назва програми (змінна класу `CProgram`).
`string` type – тип зловмисного ПО (змінна класу `CMalware`).

2.3. Опис методів

`virtual string` getInfo() `const` – виведення даних елемента у консоль (метод класу `CProgram`).
`virtual stringstream` getStr() `const` – отримання строки з даними елемента (метод класу `CProgram`).
`int` getID() `const` – отримання індекса елемента (метод класу `CProgram`).
`bool` elementOutput(`int`, `string`) – виведення елемента за обраним критерієм (метод класу `CProgram`).
`int` countElement(`int`, `string`) – виведення кількості елементів за обраним критерієм (метод класу `CProgram`).
`CProgram()` – конструктор класа за замовчуванням (метод класу `CProgram`).
`CProgram(bool, int, int, int, int, string)` – конструктор класа з параметрами (метод класу `CProgram`).
`CProgram(const CProgram&)` – конструктор копіювання (метод класу `CProgram`).
`virtual ~CProgram()` – деструктор класа (метод класу `CProgram`).
`friend ostream& operator<< (ostream&, const CProgram&)` – перевантаження оператора `<<` (метод класу `CProgram`).
`virtual bool operator==(const int) const` – перевантаження оператора `==` (метод класу `CProgram`).
`bool operator()(const shared_ptr<CProgram>&, const shared_ptr<CProgram>&)` – перевантаження оператора `()` (метод класу `Functor`).

3. Текст програми

main.cpp

```
#include "malware.h"
#include "Functor.h"

CProgram* newProgram(int);
void VectorMenu();
void ListMenu();
void MapMenu();
void SetMenu();

vector <shared_ptr<CProgram>> CombineVectors(vector<shared_ptr<CProgram>>&, vector
<shared_ptr<CProgram>>&);
map <int, shared_ptr<CProgram>> CombineMaps(map<int, shared_ptr<CProgram>>&, map<int,
shared_ptr<CProgram>>&);

int main()
{
    setlocale(LC_ALL, "Rus");
    int choise = 0;
    bool stop = 1;
```

```

while (stop)
{
    cout << "Выберите STL контейнер:" << endl;
    cout << "1. Vector" << endl;
    cout << "2. List" << endl;
    cout << "3. Map" << endl;
    cout << "4. Set" << endl;
    cout << "5. Выход" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise;

    switch (choise)
    {
        case 1:
            VectorMenu();
            break;

        case 2:
            ListMenu();
            break;

        case 3:
            MapMenu();
            break;

        case 4:
            SetMenu();
            break;

        case 5:
            stop = 0;
            break;

        default:
            cout << "Ошибка. Неверная команда. Повторите попытку." << endl;
    }
}

if (_CrtDumpMemoryLeaks())
    cout << endl << "Есть утечка памяти." << endl;
else
    cout << endl << "Утечка памяти отсутствует." << endl;

return 0;
}

CProgram* newProgram(int value)
{
    if (value % 2 == 0)
    {
        CProgram* temp = new CMalware(1, 5231, 505, 101, 56234, "KeySaver", "Keylogger");
        return temp;
    }
    else
    {
        CProgram* temp = new CProgram(0, 645, 634, 6745, 45678, "Photoshop");
        return temp;
    }
}

vector <shared_ptr<CProgram>> CombineVectors(vector<shared_ptr<CProgram>>& first, vector
<shared_ptr<CProgram>>& second)
{
    vector <shared_ptr<CProgram>> resultVector;

    resultVector.insert(resultVector.end(), make_move_iterator(first.begin()),
make_move_iterator(first.end()));
}

```

```

        resultVector.insert(resultVector.end(), make_move_iterator(second.begin()),
make_move_iterator(second.end()));

        cout << endl << "Векторы объединены." << endl;
        return resultVector;
}
map<int, shared_ptr<CProgram>> CombineMaps(map<int, shared_ptr<CProgram>>& firstMap, map<int,
shared_ptr<CProgram>>& secondMap)
{
    map<int, shared_ptr<CProgram>> resultMap;
    vector<shared_ptr<CProgram>> data1;
    vector<shared_ptr<CProgram>> data2;
    vector<int> map1Keys;
    vector<int> map2Keys;
    vector<int> temp;
    vector<int> res;

    for (auto const& it : firstMap)
        map1Keys.push_back(it.first);
    for (auto const& it : secondMap)
        map2Keys.push_back(it.first);

    for (auto const& it : firstMap)
        data1.push_back(it.second);
    for (auto const& it : secondMap)
        data2.push_back(it.second);

    sort(map1Keys.begin(), map1Keys.end());
    sort(map2Keys.begin(), map2Keys.end());

    set_intersection(map1Keys.begin(), map1Keys.end(), map2Keys.begin(), map2Keys.end(),
back_inserter(res));

    temp.insert(temp.end(), map1Keys.begin(), map1Keys.end());
    temp.insert(temp.end(), map2Keys.begin(), map2Keys.end());
    sort(temp.begin(), temp.end());
    temp.erase(unique(temp.begin(), temp.end()), temp.end());

    auto it1 = map1Keys.begin();
    auto it2 = map2Keys.begin();
    stringstream ss1, ss2;
    int count1, count2;

    int time1, time2;
    string internetTF1, internetTF2;
    bool internet;
    int size1, size2;
    int lines1, lines2;
    int index1, index2, index3;
    string name1, name2;
    string type1, type2;

    string value1, value2;

    for (size_t i = 0; i < temp.size(); i++)
    {
        if (find(res.begin(), res.end(), i + 1) != res.end())
        {
            auto itIndex1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
            auto itIndex2 = find(map2Keys.begin(), map2Keys.end(), i + 1);

            index1 = distance(map1Keys.begin(), itIndex1);
            index2 = distance(map2Keys.begin(), itIndex2);

            ss1 = data1[index1]->getStr();
            ss2 = data2[index2]->getStr();

            value1 = ss1.str();
            value2 = ss2.str();

```

```

count1 = count(value1.begin(), value1.end(), ' ');
count2 = count(value2.begin(), value2.end(), ' ');

ss1 >> name1;
ss1 >> index1;
ss1 >> time1;
ss1 >> size1;
ss1 >> lines1;
ss1 >> internetTF1;
if (count1 == 6)
    ss1 >> type1;

ss2 >> name2;
ss2 >> index2;
ss2 >> time2;
ss2 >> size2;
ss2 >> lines2;
ss2 >> internetTF2;
if (count2 == 6)
    ss2 >> type2;

name1 += name2;
index1 += index2;
time1 += time2;
size1 += size2;
lines1 += lines2;
if (internetTF1 == "1" || internetTF2 == "1")
    internet = true;
if (count1 == 6 || count2 == 6)
{
    type1 += type2;
    resultMap.emplace(i + 1, new CMalware(internet, time1, size1, lines1,
index1, name1, type1));
}
else
    resultMap.emplace(i + 1, new CProgram(internet, time1, size1, lines1,
index1, name1));
}
else
{
    it1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
    if (it1 != map1Keys.end())
    {
        index3 = distance(map1Keys.begin(), it1);
        resultMap.emplace(i + 1, data1[index3]);
    }
    else
    {
        it2 = find(map2Keys.begin(), map2Keys.end(), i + 1);
        index3 = distance(map2Keys.begin(), it2);
        resultMap.emplace(i + 1, data2[index3]);
    }
}
}

map1Keys.erase(map1Keys.begin(), map1Keys.end());
map2Keys.erase(map2Keys.begin(), map2Keys.end());
temp.erase(temp.begin(), temp.end());
res.erase(res.begin(), res.end());
data1.erase(data1.begin(), data1.end());
data2.erase(data2.begin(), data2.end());

return resultMap;
}
void VectorMenu()
{
    vector <shared_ptr<CProgram>> vector;

```

```

std::vector <shared_ptr<CProgram>> mergeVector;
std::vector<shared_ptr<CProgram>>::iterator it;
stringstream temp;
string data;
bool stop = 1, findEl = 0;
int chose = 0, chose2 = 0, chose3 = 0;
int value = 0, number = 0, result = 0, sum = 0;

for (size_t i = 0; i < 4; i++)
{
    if (i == 0)
        vector.emplace_back(new CProgram());
    else if (i == 1)
        vector.emplace_back(new CMalware(1, 8800, 555, 35, 35634, "BestMalware",
"Exploit"));
    else if (i == 2)
        vector.emplace_back(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
    else if (i == 3)
        vector.emplace_back(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer",
"Rootkit"));
}

while (stop != 0)
{
    if (vector.size() == 0)
    {
        cout << "Вектор пуст. Что вы хотите сделать?" << endl;
        cout << "1) Добавить элемент" << endl;
        cout << "2) Завершение работы" << endl;
        cout << "=====" << endl;
        cout << "Ваш выбор: ";
        cin >> chose;
        cout << endl;

        switch (chose)
        {
            case 1:
                cout << "Выберите программу, которую хотите добавить:" << endl;
                cout << "1. Элемент класса CProgram" << endl;
                cout << "2. Элемент класса CMalware" << endl;
                cout << "=====" << endl;
                cout << "Ваш выбор: ";
                cin >> value;

                try
                {
                    vector.at(value);

                    if (value == 1 || value == 2)
                    {
                        vector.emplace_back(newProgram(value));
                        cout << "Элемент добавлен." << endl;
                    }
                    else
                        cout << "Ошибка. Неверный номер." << endl;
                }
                catch (const std::exception& ex)
                {
                    cout << ex.what() << endl;
                }

                break;

            case 2:
                cout << "Завершение работы." << endl;
                stop = 0;
                break;

            default:

```

```

        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
    }
}
else
{
    cout << endl;
    cout << "1)Вывод на экран" << endl;
    cout << "2)Удаление" << endl;
    cout << "3)Добавление" << endl;
    cout << "4)Объединить векторы" << endl;
    cout << "5)Сортировка" << endl;
    cout << "6)Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{
case 1:
    cout << "Выберите команду:" << endl;
    cout << "1) Вывести весь список на экран" << endl;
    cout << "2) Вывести программу по ID" << endl;
    cout << "3) Вывести количество элементов по критерию" << endl;
    cout << "4) Найти элемент по критерию" << endl;
    cout << "5) Вернуться к выбору действий" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    switch (choise2)
    {
case 1:
        cout << setw(12) << "Название" << setw(14) << "Индекс";
        cout << setw(14) << "Время работы" << setw(8) << "Размер";
        cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
        cout << setw(10) << "Тип" << endl;

        number = 1;
        for_each(vector.begin(), vector.end(), [&number](const
shared_ptr<CProgram>& program)
        {
            cout << number << ". " << *program << endl;
            number++;
        });
        number = 1;
        break;

case 2:
        cout << "Введите id элемента, которого вы хотите получить: ";
        cin >> value;
        cout << endl;

        findEl = 0, number = -1;
        for (const auto& element: vector)
        {
            if (element->getID() == value)
            {
                number++;
                findEl = 1;
                break;
            }
            else
                number++;
        }
    }
}

```

```

        if (findEl)
        {
            temp = vector[number]->getStr();
            data = temp.str();
            cout << "Ваш элемент: " << endl;
            cout << data << endl << endl;
        }
        else
            cout << "Элемент с таким ID не найден." << endl;

        break;
    case 3:
        cout << "Выберите критерий, по которому надо искать: " << endl;
        cout << "1) Название" << endl;
        cout << "2) Время работы" << endl;
        cout << "3) Размер" << endl;
        cout << "4) Количество строк кода" << endl;
        cout << "5) Индекс" << endl;
        cout << "6) Использует ли интернет" << endl;
        cout << "7) Вернуться назад" << endl;
        cout << "===== " << endl;
        cout << "Ваш выбор: ";
        cin >> choise3;
        cout << endl;

        if (choise3 < 1 || choise3 >= 7)
        {
            cout << "Возвращение назад." << endl;
            break;
        }

        it = vector.begin();

        cout << "Введите критерий: ";
        cin.ignore();
        getline(cin, data);
        number = 0, value = 0;

        while (number < vector.size())
        {
            result = (*it)->countElement(choise3, data);
            number++;
            it++;
            sum += result;
        }
        if (sum != 0)
            cout << "Количество элементов с данным параметром: " << sum <<

endl;

        break;
    case 4:
        cout << "Выберите критерий, по которому надо искать: " << endl;
        cout << "1) Название" << endl;
        cout << "2) Время работы" << endl;
        cout << "3) Размер" << endl;
        cout << "4) Количество строк кода" << endl;
        cout << "5) Индекс" << endl;
        cout << "6) Использует ли интернет" << endl;
        cout << "7) Вернуться назад" << endl;
        cout << "===== " << endl;
        cout << "Ваш выбор: ";
        cin >> choise3;
        cout << endl;

        if (choise3 < 1 || choise3 >= 7)
        {
            cout << "Возвращение назад." << endl;
            break;
        }

```



```

    }

    it = vector.begin();
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < vector.size())
    {
        result = (*it)->elementOutput(choise3, data);
        number++;
        it++;
    }

    break;
case 5:
    cout << "Возвращение назад." << endl;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;

}
break;

case 2:
    cout << "Введите ID элемента, который хотите удалить: ";
    cin >> value;
    cout << endl;

    findEl = 0, number = -1;
    for (const auto& element:vector)
    {
        if (element->getID() == value)
        {
            number++;
            findEl = 1;
            break;
        }
        else
            number++;
    }

    if (findEl)
    {
        it = vector.begin();
        advance(it, number);
        vector.erase(it);

        cout << "Удаление выполнено." << endl;
    }
    else
        cout << "Элемент не найден." << endl;

    break;

case 3:
    cout << "Выберите программу, которую хотите добавить:" << endl;
    cout << "1. Элемент класса CProgram" << endl;
    cout << "2. Элемент класса CMalware" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> value;

    try
    {
        vector.at(value);
    }

```

```

        if (value == 1 || value == 2)
        {
            vector.emplace_back(newProgram(value));
            cout << "Элемент добавлен." << endl;
        }
        else
            cout << "Ошибка. Неверный номер." << endl;
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
    }

    break;
case 4:

    for (size_t j = 0; j < 4; j++)
    {
        if (j == 0)
            mergeVector.emplace_back(new CProgram(1, 534, 164, 4123, 56789,
"Browser"));
        else if (j == 1)
            mergeVector.emplace_back(new CProgram(1, 423, 6452, 3122, 53425,
"TextEditor"));
        else if (j == 2)
            mergeVector.emplace_back(new CMalware(0, 231, 534, 634, 23567,
"JustMap", "Worm"));
        else if (j == 3)
            mergeVector.emplace_back(new CMalware(1, 123, 345, 964, 90346,
"Imtired", "Adware"));
    }

    cout << "Контейнер, с которым будет объединение:" << endl << endl;
    cout << setw(12) << "Название" << setw(14) << "Индекс";
    cout << setw(14) << "Время работы" << setw(8) << "Размер";
    cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
    cout << setw(10) << "Тип" << endl;

    number = 1;
    for_each(mergeVector.begin(), mergeVector.end(), [&number](const
shared_ptr<CProgram>& program)
    {
        cout << number << ". " << *program << endl;
        number++;
    });
    number = 1;

    vector = CombineVectors(vector, mergeVector);
    mergeVector.erase(mergeVector.begin(), mergeVector.end());

    break;
case 5:
    cout << "Сортировать по: " << endl;
    cout << "1) Возрастанию" << endl;
    cout << "2) Убыванию" << endl;
    cout << "3) Вернуться назад" << endl;
    cout << "=====" << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    if (choise2 == 1 || choise2 == 2)
    {
        Functor funct(choise2);

        sort(vector.begin(), vector.end(), funct);
    }

```

```

        cout << "Вектор отсортирован." << endl;
    }
    else if (choise2 == 3)
        cout << "Возвращение назад." << endl;
    else
        cout << "Ошибка. Неверная команда." << endl;

    break;

case 6:
    cout << "Завершение работы." << endl << endl;
    stop = 0;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;
}
}
}
void ListMenu()
{
    list <shared_ptr<CProgram>> list;
    stringstream temp;
    string data;
    bool stop = 1, findEl = 0;
    int choise = 0, choise2 = 0, choise3 = 0;
    int value = 0;
    int number = 0;
    int result = 0, sum = 0;
    auto it = list.begin();

    for (size_t i = 0; i < 4; i++)
    {
        if (i == 0)
            list.emplace_back(new CProgram());
        else if (i == 1)
            list.emplace_back(new CMalware(1, 8800, 555, 35, 35634, "BestMalware",
"Exploit"));
        else if (i == 2)
            list.emplace_back(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
        else if (i == 3)
            list.emplace_back(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer",
"Rootkit"));
    }

    while (stop != 0)
    {
        if (list.size() == 0)
        {
            cout << "Вектор пуст. Что вы хотите сделать?" << endl;
            cout << "1) Добавить элемент" << endl;
            cout << "2) Завершение работы" << endl;
            cout << "=====" << endl;
            cout << "Ваш выбор: ";
            cin >> choise;
            cout << endl;

            switch (choise)
            {
            case 1:
                cout << "Выберите программу, которую хотите добавить:" << endl;
                cout << "1. Элемент класса CProgram" << endl;
                cout << "2. Элемент класса CMalware" << endl;
                cout << "=====" << endl;
                cout << "Ваш выбор: ";
                cin >> value;

```

```

        try
        {
            if (value == 1 || value == 2)
            {
                list.emplace_front(newProgram(value));
                cout << "Элемент добавлен." << endl;
            }
            else
                cout << "Ошибка. Неверный номер." << endl;
        }
        catch (const std::exception & ex)
        {
            cout << ex.what() << endl;
        }

        break;

    case 2:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;

    default:
        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
}
else
{
    cout << endl;
    cout << "1)Вывод на экран" << endl;
    cout << "2)Удаление элемента" << endl;
    cout << "3)Добавление элементов" << endl;
    cout << "4)Сортировка элементов" << endl;
    cout << "5)Завершение работы" << endl;
    cout << "=====" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{
    case 1:
        cout << "Выберите команду:" << endl;
        cout << "1) Вывести весь список на экран" << endl;
        cout << "2) Вывести программу по ID" << endl;
        cout << "3) Вывести количество элементов по критерию" << endl;
        cout << "4) Найти элемент по критерию" << endl;
        cout << "5) Вернуться к выбору действий" << endl;
        cout << "=====" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;

        switch (choise2)
        {
            case 1:
                cout << setw(12) << "Название" << setw(14) << "Индекс";
                cout << setw(14) << "Время работы" << setw(8) << "Размер";
                cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
                cout << setw(10) << "Тип" << endl;

                number = 1;
                for_each(list.begin(), list.end(), [&number](const shared_ptr<CProgram>&
program)
                    {
                        cout << number << ". " << *program << endl;
                        number++;
                    }
                );
            }
        }
    }
}

```

```

        });
        number = 1;
        break;

case 2:
    cout << "Введите id элемента, которого вы хотите получить: ";
    cin >> value;
    cout << endl;

    findEl = 0, number = -1;
    for (const auto& element : list)
    {
        if (element->getID() == value)
        {
            number++;
            findEl = 1;
            break;
        }
        else
            number++;
    }

    if (findEl)
    {
        it = list.begin();
        advance(it, number);

        temp = (*it)->getStr();
        data = temp.str();

        cout << "Ваш элемент: " << endl;
        cout << data << endl << endl;
    }
    else
        cout << "Элемент с таким ID не найден." << endl;

    break;

case 3:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 < 1 || choise3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

    it = list.begin();
    result = 0, sum = 0;
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < list.size())
    {
        result = (*it)->countElement(choise3, data);
        number++;
    }

```

```

        it++;
        sum += result;
    }
    if (sum != 0)
        cout << "Количество элементов с данным параметром: " << sum <<

endl;

    break;

case 4:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 < 1 || choise3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

    it = list.begin();
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < list.size())
    {
        result = (*it)->elementOutput(choise3, data);
        number++;
        it++;
    }

    break;
case 5:
    cout << "Возвращение назад." << endl;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;

}
break;

case 2:
    cout << "Введите ID элемента, который хотите удалить: ";
    cin >> value;
    cout << endl;

    findEl = 0, number = -1;
    for (const auto& element : list)
    {
        if (element->getID() == value)
        {
            number++;
            findEl = 1;
            break;
        }
        else

```

```

        number++;
    }

    if (findEl)
    {
        it = list.begin();
        advance(it, number);
        list.erase(it);

        cout << "Удаление выполнено." << endl;
    }
    else
        cout << "Элемент не найден." << endl;

    break;

case 3:
    cout << "Выберите программу, которую хотите добавить:" << endl;
    cout << "1. Элемент класса CProgram" << endl;
    cout << "2. Элемент класса CMalware" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> value;

    try
    {
        if (value == 1 || value == 2)
        {
            list.emplace_front(newProgram(value));
            cout << "Элемент добавлен." << endl;
        }
        else
            cout << "Ошибка. Неверный номер." << endl;
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
    }

    break;

case 4:
    cout << "Сортировать по: " << endl;
    cout << "1) Возрастаю" << endl;
    cout << "2) Убыванию" << endl;
    cout << "3) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    if (choise2 == 1 || choise2 == 2)
    {
        Functor funct(choise2);

        list.sort(funct);

        cout << "Список отсортирован." << endl;
    }
    else if (choise2 == 3)
        cout << "Возвращение назад." << endl;
    else
        cout << "Ошибка. Неверная команда." << endl;

    break;

case 5:
    cout << "Завершение работы." << endl << endl;
    stop = 0;
    break;

```

```

        default:
            cout << "Неверный символ. Повторите попытку." << endl;
            break;
        }
    }
}

void MapMenu()
{
    map <int, shared_ptr<CProgram>> map;
    stringstream temp;
    string data;
    bool stop = 1, findEl = 0;
    int chose = 0, chose2 = 0, chose3 = 0;
    int value = 0;
    int i = 0;
    int number = 0, sum = 0, result = 0;
    auto it = map.begin();

    for (; i < 4; i++)
    {
        if (i == 0)
            map.emplace(i + 1, new CProgram());
        else if (i == 1)
            map.emplace(i + 1, new CMalware(1, 8800, 555, 35, 35634, "BestMalware",
"Exploit"));
        else if (i == 2)
            map.emplace(i + 1, new CProgram(0, 423, 523, 654, 53453, "Calculator"));
        else if (i == 3)
            map.emplace(i + 1, new CMalware(0, 345, 789, 423, 67456, "MoneyStealer",
"Rootkit"));
    }

    while (stop != 0)
    {
        if (map.size() == 0)
        {
            cout << "Вектор пуст. Что вы хотите сделать?" << endl;
            cout << "1) Добавить элемент" << endl;
            cout << "2) Завершение работы" << endl;
            cout << "=====" << endl;
            cout << "Ваш выбор: ";
            cin >> chose;
            cout << endl;

            switch (chose)
            {
            case 1:
                cout << "Выберите программу, которую хотите добавить:" << endl;
                cout << "1. Элемент класса CProgram" << endl;
                cout << "2. Элемент класса CMalware" << endl;
                cout << "=====" << endl;
                cout << "Ваш выбор: ";
                cin >> value;

                try
                {
                    if (value == 1 || value == 2)
                    {
                        map.emplace(++i, newProgram(value));
                        cout << "Элемент добавлен." << endl;
                    }
                    else
                        cout << "Ошибка. Неверный номер." << endl;
                }
                catch (const std::exception & ex)
                {
                    cout << ex.what() << endl;
                }
            }
        }
    }
}

```



```

        }

        break;

    case 2:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;

    default:
        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
    }
}
else
{
    cout << endl;
    cout << "1)Вывод на экран" << endl;
    cout << "2)Удаление элемента" << endl;
    cout << "3)Добавление элементов" << endl;
    cout << "4)Сортировать контейнеры" << endl;
    cout << "5)Объединить контейнеры" << endl;
    cout << "6)Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{
    case 1:
        cout << "Выберите команду:" << endl;
        cout << "1) Вывести весь список на экран" << endl;
        cout << "2) Вывести программу по ID" << endl;
        cout << "3) Вывести количество элементов по критерию" << endl;
        cout << "4) Найти элемент по критерию" << endl;
        cout << "5) Вернуться к выбору действий" << endl;
        cout << "======" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;

        switch (choise2)
        {
            case 1:
                cout << setw(12) << "Название" << setw(14) << "Индекс";
                cout << setw(14) << "Время работы" << setw(8) << "Размер";
                cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
                cout << setw(10) << "Тип" << endl;

                for_each(map.begin(), map.end(), [](const std::pair<const int,
shared_ptr<CProgram>>& program)
                {
                    cout << program.first << ". " << *program.second << endl;
                });

                break;

            case 2:
                cout << "Введите номер элемента, которого вы хотите получить: ";
                cin >> value;
                cout << endl;

                findEl = 0;
                it = map.find(value);

                if (it != map.end())
                {

```

```

        temp = (*it).second->getStr();
        data = temp.str();

        cout << "Ваш элемент: " << endl;
        cout << data << endl << endl;
    }
    else
        cout << "Элемент с таким ID не найден." << endl;

    break;

case 3:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 < 1 || choise3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

    it = map.begin();
    result = 0, sum = 0;
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < map.size())
    {
        result = it->second->countElement(choise3, data);
        number++;
        it++;
        sum += result;
    }
    if (sum != 0)
        cout << "Количество элементов с данным параметром: " << sum <<

endl;

    break;

case 4:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 < 1 || choise3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

```

```

    }

    it = map.begin();
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < map.size())
    {
        result = it->second->elementOutput(choise3, data);
        number++;
        it++;
    }

    break;
case 5:
    cout << "Возвращение назад." << endl;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;

}
break;

case 2:
    cout << "Введите номер элемента, который хотите удалить: ";
    cin >> value;
    cout << endl;

    findEl = 0;
    it = map.find(value);

    if (it != map.end())
    {
        map.erase(it);
        cout << "Удаление выполнено." << endl;
    }
    else
        cout << "Элемент не найден." << endl;

    break;

case 3:
    cout << "Выберите программу, которую хотите добавить:" << endl;
    cout << "1. Элемент класса CProgram" << endl;
    cout << "2. Элемент класса CMalware" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> value;

    try
    {
        if (value == 1 || value == 2)
        {
            map.emplace(++i, newProgram(value));
            cout << "Элемент добавлен." << endl;
        }
        else
            cout << "Ошибка. Неверный номер." << endl;
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
    }

    break;

```

```

case 4:
    cout << "Сортировать по: " << endl;
    cout << "1) Возрастанию" << endl;
    cout << "2) Убыванию" << endl;
    cout << "3) Вернуться назад" << endl;
    cout << "=====" << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    if (choise2 == 1 || choise2 == 2)
    {
        value = 1;
        Functor funct(choise2);
        vector <shared_ptr<CProgram>> temp;

        for (auto const& it : map)
            temp.push_back(it.second);

        sort(temp.begin(), temp.end(), funct);

        map.erase(map.begin(), map.end());

        transform(temp.begin(), temp.end(), inserter(map, map.end()),
[&value](const shared_ptr<CProgram>& a)
        {
            return make_pair(value++, a);
        });
    }
    else if (choise2 == 3)
        cout << "Возвращение." << endl;
    else
        cout << "Ошибка. Неверный номер элемента." << endl;

    break;

case 5:
{
    std::map <int, shared_ptr<CProgram>> mergeMap;
    std::map <int, shared_ptr<CProgram>> result;
    for (size_t j = 0; j < 4; j++)
    {
        if (j == 0)
            mergeMap.emplace(j + 1, new CProgram(1, 534, 164, 4123, 56789,
"Browser"));
        else if (j == 1)
            mergeMap.emplace(j + 1, new CProgram(1, 423, 6452, 3122, 53425,
"TextEditor"));
        else if (j == 2)
            mergeMap.emplace(j + 1, new CMalware(0, 231, 534, 634, 23567,
"JustMap", "Worm"));
        else if (j == 3)
            mergeMap.emplace(j + 1, new CMalware(1, 123, 345, 964, 90346,
"Imtired", "Adware"));
    }

    cout << "Контейнер, с которым будет объединение:" << endl << endl;
    cout << setw(12) << "Название" << setw(14) << "Индекс";
    cout << setw(14) << "Время работы" << setw(8) << "Размер";
    cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
    cout << setw(10) << "Тип" << endl;
    for_each(mergeMap.begin(), mergeMap.end(), [](const pair<const int,
shared_ptr<CProgram>>& program)
    {
        cout << program.first << ". " << *program.second << endl;
    });

    map = CombineMaps(map, mergeMap);
}

```

```

        cout << "Объединение выполнено." << endl;

        break;
    }
    case 6:
        cout << "Завершение работы." << endl << endl;
        stop = 0;
        break;

    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
    }
}

void SetMenu()
{
    set <shared_ptr<CProgram>> set;
    stringstream ss;
    string data;
    bool stop = 1, findEl = 0;
    int chose = 0, chose2 = 0, chose3 = 0;
    int value = 0, number = 0, result = 0, sum = 0;
    auto it = set.begin();

    for (size_t i = 0; i < 4; i++)
    {
        if (i == 0)
            set.emplace(new CProgram());
        else if (i == 1)
            set.emplace(new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit"));
        else if (i == 2)
            set.emplace(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
        else if (i == 3)
            set.emplace(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer", "Rootkit"));
    }

    while (stop != 0)
    {
        if (set.size() == 0)
        {
            cout << "Вектор пуст. Что вы хотите сделать?" << endl;
            cout << "1) Добавить элемент" << endl;
            cout << "2) Завершение работы" << endl;
            cout << "===== " << endl;
            cout << "Ваш выбор: ";
            cin >> chose;
            cout << endl;

            switch (chose)
            {
            case 1:
                cout << "Выберите программу, которую хотите добавить:" << endl;
                cout << "1. Элемент класса CProgram" << endl;
                cout << "2. Элемент класса CMalware" << endl;
                cout << "===== " << endl;
                cout << "Ваш выбор: ";
                cin >> value;

                try
                {
                    if (value == 1 || value == 2)
                    {
                        set.emplace(newProgram(value));
                        cout << "Элемент добавлен." << endl;
                    }
                    else

```

```

        cout << "Ошибка. Неверный номер." << endl;
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
    }

    break;

case 2:
    cout << "Завершение работы." << endl;
    stop = 0;
    break;

default:
    cout << "Неверный номер элемента. Повторите попытку." << endl;
    break;
}
}
else
{
    cout << endl;
    cout << "1)Вывод на экран" << endl;
    cout << "2)Удаление элемента" << endl;
    cout << "3)Добавление элементов" << endl;
    cout << "4)Сортировка элементов" << endl;
    cout << "5)Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{
case 1:
    cout << "Выберите команду:" << endl;
    cout << "1) Вывести весь список на экран" << endl;
    cout << "2) Вывести программу по ID" << endl;
    cout << "3) Вывести количество элементов по критерию" << endl;
    cout << "4) Найти элемент по критерию" << endl;
    cout << "5) Вернуться к выбору действий" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    switch (choise2)
    {
case 1:
        cout << setw(12) << "Название" << setw(14) << "Индекс";
        cout << setw(14) << "Время работы" << setw(8) << "Размер";
        cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
        cout << setw(10) << "Тип" << endl;

        number = 1;
        for_each(set.begin(), set.end(), [&number](const shared_ptr<CProgram>&
program)
        {
            cout << number << ". " << *program << endl;
            number++;
        });
        number = 1;
        break;

case 2:
        cout << "Введите id элемента, которого вы хотите получить: ";
        cin >> value;
        cout << endl;

```

```

findEl = 0, number = -1;
for (const auto& element : set)
{
    if (element->getID() == value)
    {
        number++;
        findEl = 1;
        break;
    }
    else
        number++;
}

if (findEl)
{
    it = set.begin();
    advance(it, number);

    ss = (*it)->getStr();
    data = ss.str();

    cout << "Ваш элемент: " << endl;
    cout << data << endl << endl;
}
else
    cout << "Элемент с таким ID не найден." << endl;

break;

case 3:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choise3;
    cout << endl;

    if (choise3 < 1 || choise3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

    it = set.begin();
    result = 0, sum = 0;
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < set.size())
    {
        result = (*it)->countElement(choise3, data);
        number++;
        it++;
        sum += result;
    }
    if (sum != 0)
        cout << "Количество элементов с данным параметром: " << sum <<
endl;

break;

```

```

case 4:
    cout << "Выберите критерий, по которому надо искать: " << endl;
    cout << "1) Название" << endl;
    cout << "2) Время работы" << endl;
    cout << "3) Размер" << endl;
    cout << "4) Количество строк кода" << endl;
    cout << "5) Индекс" << endl;
    cout << "6) Использует ли интернет" << endl;
    cout << "7) Вернуться назад" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choice3;
    cout << endl;

    if (choice3 < 1 || choice3 >= 7)
    {
        cout << "Возвращение назад." << endl;
        break;
    }

    it = set.begin();
    cout << "Введите критерий: ";
    cin.ignore();
    getline(cin, data);
    number = 0, value = 0;

    while (number < set.size())
    {
        result = (*it)->elementOutput(choice3, data);
        number++;
        it++;
    }

    break;
case 5:
    cout << "Возвращение назад." << endl;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;

}
break;

case 2:
    cout << "Введите ID элемента, который хотите удалить: ";
    cin >> value;
    cout << endl;

    findEl = 0, number = -1;
    for (const auto& element : set)
    {
        if (element->getID() == value)
        {
            number++;
            findEl = 1;
            break;
        }
        else
            number++;
    }

    if (findEl)
    {
        it = set.begin();
        advance(it, number);
        set.erase(it);
    }

```



```

        cout << "Удаление выполнено." << endl;
    }
    else
        cout << "Элемент не найден." << endl;

    break;

case 3:
    cout << "Выберите программу, которую хотите добавить:" << endl;
    cout << "1. Элемент класса CProgram" << endl;
    cout << "2. Элемент класса CMalware" << endl;
    cout << "=====" << endl;
    cout << "Ваш выбор: ";
    cin >> value;

    try
    {
        if (value == 1 || value == 2)
        {
            set.emplace(newProgram(value));
            cout << "Элемент добавлен." << endl;
        }
        else
            cout << "Ошибка. Неверный номер." << endl;
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
    }

    break;

case 4:
    cout << "Сортировать по: " << endl;
    cout << "1) Возрастанию" << endl;
    cout << "2) Убыванию" << endl;
    cout << "3) Вернуться назад" << endl;
    cout << "=====" << endl;
    cout << "Ваш выбор: ";
    cin >> choise2;
    cout << endl;

    if (choise2 == 1 || choise2 == 2)
    {
        vector <shared_ptr<CProgram>> temp(set.begin(), set.end());
        set.erase(set.begin(), set.end());

        Functor funct(choise2);
        sort(temp.begin(), temp.end(), funct);

        set.insert(temp.begin(), temp.end());

        cout << "Отсортированный set" << endl;
        cout << setw(12) << "Название" << setw(14) << "Индекс";
        cout << setw(14) << "Время работы" << setw(8) << "Размер";
        cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
        cout << setw(10) << "Тип" << endl;

        number = 1;
        for_each(temp.begin(), temp.end(), [&number](const shared_ptr<CProgram>&
program)
        {
            cout << number << ". " << *program << endl;
            number++;
        });
        number = 1;
        temp.erase(temp.begin(), temp.end());
    }
    else if (choise2 == 3)

```

```

        cout << "Возвращение." << endl;
    else
        cout << "Ошибка. Неверный номер элемента." << endl;

    break;

case 5:
    cout << "Завершение работы." << endl << endl;
    stop = 0;
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;
}
}
}

```

malware.cpp

```

#include "malware.h"
stringstream CMalware::getStr() const
{
    stringstream temp;

    temp << name << " " << index << " " << timeOfWork
        << " " << size << " " << amountOfLines << " "
        << useInternet << " " << type;

    return temp;
}
string CMalware::getInfo() const
{
    stringstream temp;

    temp.setf(ios::left);
    temp << setw(18) << name << setw(12) << index
        << setw(11) << timeOfWork << setw(13) << size
        << setw(12) << amountOfLines << setw(12) << boolalpha << useInternet
        << setw(14) << type;

    return temp.str();
}
int CMalware::countElement(int value, string data)
{
    try
    {
        if (value == 1)
        {
            if (this->name == data)
                return 1;
            else
                return 0;
        }
        else if (value == 2)
        {
            int number = stoi(data);
            if (this->timeOfWork == number)
                return 1;
            else
                return 0;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)

```

```

        return 1;
    else
        return 0;
}
else if (value == 4)
{
    int number = stoi(data);
    if (this->amountOfLines == number)
        return 1;
    else
        return 0;
}
else if (value == 5)
{
    int number = stoi(data);
    if (this->index == number)
        return 1;
    else
        return 0;
}
else if (value == 6)
{
    int number = 0;
    if (data == "true" || data == "true" || data == "1")
        number = 1;
    else
        number = 0;

    if (this->useInternet == number)
        return 1;
    else
        return 0;
}
else if (value == 7)
{
    if (this->type == data)
        return 1;
    else
        return 0;
}
}
catch (const std::exception & ex)
{
    cout << ex.what() << endl;
    return 0;
}

return 0;
}
bool CMalware::elementOutput(int value, string data)
{
    try
    {
        if (value == 1)
        {
            if (this->name == data)
                cout << *this << endl;
            return true;
        }
        else if (value == 2)
        {
            int number = stoi(data);
            if (this->timeOfWork == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 3)
        {
            int number = stoi(data);

```

```

        if (this->size == number)
            cout << *this << endl;
        return true;
    }
    else if (value == 4)
    {
        int number = stoi(data);
        if (this->amountOfLines == number)
            cout << *this << endl;
        return true;
    }
    else if (value == 5)
    {
        int number = stoi(data);
        if (this->index == number)
            cout << *this << endl;
        return true;
    }
    else if (value == 6)
    {
        int number = 0;
        if (data == "true" || data == "true" || data == "1")
            number = 1;
        else
            number = 0;

        if (this->useInternet == number)
            return 1;
        else
            return 0;
    }
    else if (value == 7)
    {
        if (this->type == data)
            cout << *this << endl;
        return true;
    }
}
catch (const std::exception & ex)
{
    cout << ex.what() << endl;
    return 0;
}

return 0;
}

```

```

CMalware::CMalware(bool internet, int time, int size, int lines, int index, string name, string
type) : CProgram(internet, time, size, lines, index, name), type(type) {}
CMalware::CMalware() : CProgram(), type("Exploit") {}
CMalware::CMalware(const CMalware& other) : CProgram(other), type(other.type) {}
CMalware::~CMalware() {}

```

```

bool CMalware::operator==(const int id) const
{
    return this->index == id;
}

```

program.cpp

```

#include "program.h"

string CProgram::getInfo() const
{
    stringstream temp;

    temp.setf(std::ios::left);

```

```

temp << setw(18) << name << setw(12) << index << setw(11)
    << timeOfWork << setw(13) << size << setw(12)
    << amountOfLines << setw(8) << boolalpha << useInternet;

return temp.str();
}
int CProgram::getID() const
{
    return index;
}
stringstream CProgram::getStr() const
{
    stringstream temp;
    temp << name << " " << index << " " << timeOfWork << " "
        << size << " " << amountOfLines << " " << useInternet;

    return temp;
}
int CProgram::countElement(int value, string data)
{
    try
    {
        if (value == 1)
        {
            if (this->name == data)
                return 1;
            else
                return 0;
        }
        else if (value == 2)
        {
            int number = stoi(data);
            if (this->timeOfWork == number)
                return 1;
            else
                return 0;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)
                return 1;
            else
                return 0;
        }
        else if (value == 4)
        {
            int number = stoi(data);
            if (this->amountOfLines == number)
                return 1;
            else
                return 0;
        }
        else if (value == 5)
        {
            int number = stoi(data);
            if (this->index == number)
                return 1;
            else
                return 0;
        }
        else if (value == 6)
        {
            int number = 0;
            if (data == "true" || data == "true" || data == "1")
                number = 1;
            else
                number = 0;
        }
    }
}

```

```

        if (this->useInternet == number)
            return 1;
        else
            return 0;
    }
}
catch (const std::exception& ex)
{
    cout << ex.what() << endl;
    return 0;
}

return 0;
}

bool CProgram::elementOutput(int value, string data)
{
    try
    {
        if (value == 1)
        {
            if (this->name == data)
                cout << *this << endl;
            return true;
        }
        else if (value == 2)
        {
            int number = stoi(data);
            if (this->timeOfWork == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 4)
        {
            int number = stoi(data);
            if (this->amountOfLines == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 5)
        {
            int number = stoi(data);
            if (this->index == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 6)
        {
            int number = 0;
            if (data == "true" || data == "true" || data == "1")
                number = 1;
            else
                number = 0;

            if (this->useInternet == number)
                return 1;
            else
                return 0;
        }
    }
    catch (const std::exception& ex)
    {
        cout << ex.what() << endl;
    }
}

```

```

        return 0;
    }

    return 0;
}

ostream& operator<< (ostream& output, const CProgram& program)
{
    output << program.getInfo();
    return output;
}

bool CProgram::operator==(const int id) const
{
    return this->index == id;
}

CProgram::CProgram(bool internet, int time, int size, int lines, int index, string name) :
useInternet(internet), timeOfWork(time), size(size), amountOfLines(lines), index(index), name(name)
{}
CProgram::CProgram() : useInternet(false), timeOfWork(0), size(0), amountOfLines(0), index(0101),
name("Basic") {}
CProgram::CProgram(const CProgram& other) : useInternet(other.useInternet),
timeOfWork(other.timeOfWork), size(other.size), amountOfLines(other.amountOfLines),
index(other.index), name(other.name) {}
CProgram::~CProgram() {}

```

test.cpp

```

void VectorTest();
void ListTest();
void MapTest();
void SetTest();

vector <shared_ptr<CProgram>> CombineVectors(vector<shared_ptr<CProgram>>&, vector
<shared_ptr<CProgram>>&);
map <int, shared_ptr<CProgram>> CombineMaps(map<int, shared_ptr<CProgram>>&, map<int,
shared_ptr<CProgram>>&);

int main()
{
    setlocale(LC_ALL, "Rus");

    VectorTest();
    ListTest();
    MapTest();
    SetTest();

    if (_CrtDumpMemoryLeaks())
        cout << "\nЕсть утечка памяти.\n";
    else
        cout << "\nУтечка памяти отсутствует.\n";

    return 0;
}

void VectorTest()
{
    cout << "Vector" << endl;
    vector <shared_ptr<CProgram>> vector;
    std::vector <shared_ptr<CProgram>> vectorMerge;
    std::vector<shared_ptr<CProgram>>::const_iterator it;
    Functor funct(1);
    stringstream line;
    string data;
    int vectorSize;
    int value, result = 0, sum = 0;
    int i = 0;
}

```

```

for (size_t i = 0; i < 4; i++)
{
    if (i == 0)
        vector.emplace_back(new CProgram());
    else if (i == 1)
        vector.emplace_back(new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit"));
    else if (i == 2)
        vector.emplace_back(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
    else if (i == 3)
        vector.emplace_back(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer", "Rootkit"));
}

vectorSize = vector.size();
vector.emplace_back(new CMalware());
if(vectorSize != vector.size())
    cout << "Тест добавления элемента\tвыполнен успешно.\n";
else
    cout << "Тест добавления элемента\tне выполнен успешно.\n";

it = vector.begin();
advance(it, 2);
vector.erase(it);
if (vectorSize == vector.size())
    cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

line = vector[0]->getStr();
data = line.str();
if (data == "Basic 65 0 0 0 0")
    cout << "Тест получения элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест получения элемента\t\tне выполнен успешно.\n";

it = vector.begin();
data = "false";
while (i < vector.size())
{
    result = (*it)->countElement(6, data);
    i++;
    it++;
    sum += result;
}
if (sum == 3)
    cout << "Тест подсчёта элементов\t\tвыполнен успешно.\n";
else
    cout << "Тест подсчёта элементов\t\tне выполнен успешно.\n";

sort(vector.begin(), vector.end(), funct);
line = vector[0]->getStr();
data = line.str();
if(data == "Basic 65 0 0 0 0")
    cout << "Тест сортировки вектора\t\tвыполнен успешно.\n";
else
    cout << "Тест сортировки вектора\t\tне выполнен успешно.\n";

vectorSize = vector.size();
vectorMerge = vector;
vector = CombineVectors(vector, vectorMerge);
if (vector.size() != vectorSize && vector.size() == (vectorSize + vectorMerge.size()))
    cout << "Тест слияния векторов\t\tвыполнен успешно.\n";
else
    cout << "Тест слияния векторов\t\tне выполнен успешно.\n";
}
void ListTest()
{
    cout << endl << "List" << endl;
}

```



```

list <shared_ptr<CProgram>> list;
std::list<shared_ptr<CProgram>>::const_iterator it;
Functor funct(1);
int listSize;
int value, sum = 0, result = 0;
int i = 0;
stringstream line;
string data;

for (size_t i = 0; i < 4; i++)
{
    if (i == 0)
        list.emplace_back(new CProgram());
    else if (i == 1)
        list.emplace_back(new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit"));
    else if (i == 2)
        list.emplace_back(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
    else if (i == 3)
        list.emplace_back(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer", "Rootkit"));
}

listSize = list.size();
list.emplace_back(new CMalware());
if (listSize < list.size())
    cout << "Тест добавления элемента\tвыполнен успешно.\n";
else
    cout << "Тест добавления элемента\tне выполнен успешно.\n";

it = list.begin();
list.erase(it);
if (list.size() == listSize)
    cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

it = list.begin();
line = (*it)->getStr();
data = line.str();
if(data == "BestMalware 35634 8800 555 35 1 Exploit")
    cout << "Тест получения элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест получения элемента\t\tне выполнен успешно.\n";

it = list.begin();
data = "false";
while (i < list.size())
{
    result = (*it)->countElement(6, data);
    i++;
    it++;
    sum += result;
}
if (sum == 3)
    cout << "Тест подсчёта элементов\t\tвыполнен успешно.\n";
else
    cout << "Тест подсчёта элементов\t\tне выполнен успешно.\n";

list.sort(funct);
it = list.begin();
line = (*it)->getStr();
data = line.str();
if (data == "Basic 65 0 0 0 0 Exploit")
    cout << "Тест сортировки вектора\t\tвыполнен успешно.\n";
else
    cout << "Тест сортировки вектора\t\tне выполнен успешно.\n";
}
void SetTest()
{

```

```

cout << endl << "Set" << endl;
set <shared_ptr<CProgram>> set;
std::set<shared_ptr<CProgram>>::const_iterator it;
Functor funct(1);
stringstream line;
string data;
int setSize, sum = 0, i = 0;
int value;

for (size_t i = 0; i < 4; i++)
{
    if (i == 0)
        set.emplace(new CProgram());
    else if (i == 1)
        set.emplace(new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit"));
    else if (i == 2)
        set.emplace(new CProgram(0, 423, 523, 654, 53453, "Calculator"));
    else if (i == 3)
        set.emplace(new CMalware(0, 345, 789, 423, 67456, "MoneyStealer", "Rootkit"));
}

setSize = set.size();
set.emplace(new CMalware());
if (setSize < set.size())
    cout << "Тест добавления элемента\tвыполнен успешно.\n";
else
    cout << "Тест добавления элемента\tне выполнен успешно.\n";

it = set.begin();
set.erase(it);
if (set.size() == setSize)
    cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

it = set.begin();
line = (*it)->getStr();
data = line.str();
if (data == "MoneyStealer 67456 345 789 423 0 Rootkit" || data == "BestMalware 35634 8800 555 35
1 Exploit" || data == "Basic 65 0 0 0 0 Exploit")
    cout << "Тест получения элемента\t\tвыполнен успешно.\n";
else
    cout << "Тест получения элемента\t\tне выполнен успешно.\n";

it = set.begin();
data = "53453";
while (i < set.size())
{
    value = (*it)->countElement(5, data);
    i++;
    it++;
    sum += value;
}
if (sum == 1)
    cout << "Тест подсчёта элементов\t\tвыполнен успешно.\n";
else
    cout << "Тест подсчёта элементов\t\tне выполнен успешно.\n";

vector <shared_ptr<CProgram>> temp(set.begin(), set.end());
set.erase(set.begin(), set.end());
sort(temp.begin(), temp.end(), funct);
set.insert(temp.begin(), temp.end());
it = set.begin();
line = (*it)->getStr();
data = line.str();
if (data == "BestMalware 35634 8800 555 35 1 Exploit" || data == "MoneyStealer 67456 345 789 423
0 Rootkit" || data == "Basic 65 0 0 0 0 Exploit")
    cout << "Тест сортировки сета\t\tвыполнен успешно.\n";
else

```

```

        cout << "Тест сортировки сета\t\tне выполнен успешно.\n";
    }
void MapTest()
{
    cout << endl << "Map" << endl;
    std::map<int, shared_ptr<CProgram>>::const_iterator it;
    std::map<int, shared_ptr<CProgram>> mergeMap;
    map<int, shared_ptr<CProgram>> map;
    vector<shared_ptr<CProgram>> temp;
    Functor funct(1);
    stringstream line;
    string data;
    int number = 0, result, sum = 0;
    int mapSize;
    int i = 0;

    for (; i < 4; i++)
    {
        if (i == 0)
            map.emplace(i + 1, new CProgram());
        else if (i == 1)
            map.emplace(i + 1, new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit"));
        else if (i == 2)
            map.emplace(i + 1, new CProgram(0, 423, 523, 654, 53453, "Calculator"));
        else if (i == 3)
            map.emplace(i + 1, new CMalware(0, 345, 789, 423, 67456, "MoneyStealer", "Rootkit"));
    }

    mapSize = map.size();
    map.emplace(++i, new CMalware);
    if (mapSize < map.size())
        cout << "Тест добавления элемента\tвыполнен успешно.\n";
    else
        cout << "Тест добавления элемента\tне выполнен успешно.\n";

    it = map.begin();
    map.erase(it);
    if (mapSize == map.size())
        cout << "Тест удаления элемента\t\tвыполнен успешно.\n";
    else
        cout << "Тест удаления элемента\t\tне выполнен успешно.\n";

    it = map.begin();
    line = it->second->getStr();
    data = line.str();
    if (data == "BestMalware 35634 8800 555 35 1 Exploit")
        cout << "Тест получения элемента\t\tвыполнен успешно.\n";
    else
        cout << "Тест получения элемента\t\tне выполнен успешно.\n";

    data = "53453";
    while (number < map.size())
    {
        result = it->second->countElement(5, data);
        number++;
        it++;
        sum += result;
    }
    if (sum == 1)
        cout << "Тест подсчёта элементов\t\tвыполнен успешно.\n";
    else
        cout << "Тест подсчёта элементов\t\tне выполнен успешно.\n";

    sort(temp.begin(), temp.end(), funct);
    transform(temp.begin(), temp.end(), inserter(map, map.end()), [&number](const
shared_ptr<CProgram>& program)
    {
        return std::make_pair(number++, program);
    });
}

```

```

    });
    it = map.begin();
    line = it->second->getStr();
    data = line.str();
    if (data == "BestMalware 35634 8800 555 35 1 Exploit")
        cout << "Тест сортировки map\t\tвыполнен успешно.\n";
    else
        cout << "Тест сортировки map\t\tне выполнен успешно.\n";

    for (size_t j = 0; j < 4; j++)
    {
        if (j == 0)
            mergeMap.emplace(j + 1, new CProgram(1, 534, 164, 4123, 56789, "Browser"));
        else if (j == 1)
            mergeMap.emplace(j + 1, new CProgram(1, 423, 6452, 3122, 53425, "TextEditor"));
        else if (j == 2)
            mergeMap.emplace(j + 1, new CMalware(0, 231, 534, 634, 23567, "JustMap", "Worm"));
        else if (j == 3)
            mergeMap.emplace(j + 1, new CMalware(1, 123, 345, 964, 90346, "Imtired", "Adware"));
    }
    map = CombineMaps(map, mergeMap);
    if (map.size() != mapSize)
        cout << "Тест слияния map\t\tвыполнен успешно.\n";
    else
        cout << "Тест слияния map\t\tне выполнен успешно.\n";
}

vector <shared_ptr<CProgram>> CombineVectors(vector<shared_ptr<CProgram>>& first, vector
<shared_ptr<CProgram>>& second)
{
    vector <shared_ptr<CProgram>> resultVector;

    resultVector.insert(resultVector.end(), make_move_iterator(first.begin()),
make_move_iterator(first.end()));
    resultVector.insert(resultVector.end(), make_move_iterator(second.begin()),
make_move_iterator(second.end()));

    cout << endl << "Векторы объединены." << endl;
    return resultVector;
}

map <int, shared_ptr<CProgram>> CombineMaps(map<int, shared_ptr<CProgram>>& firstMap, map<int,
shared_ptr<CProgram>>& secondMap)
{
    map <int, shared_ptr<CProgram>> resultMap;
    vector <shared_ptr<CProgram>> data1;
    vector <shared_ptr<CProgram>> data2;
    vector <int> map1Keys;
    vector <int> map2Keys;
    vector <int> temp;
    vector <int> res;

    for (auto const& it : firstMap)
        map1Keys.push_back(it.first);
    for (auto const& it : secondMap)
        map2Keys.push_back(it.first);

    for (auto const& it : firstMap)
        data1.push_back(it.second);
    for (auto const& it : secondMap)
        data2.push_back(it.second);

    sort(map1Keys.begin(), map1Keys.end());
    sort(map2Keys.begin(), map2Keys.end());

    set_intersection(map1Keys.begin(), map1Keys.end(), map2Keys.begin(), map2Keys.end(),
back_inserter(res));

    temp.insert(temp.end(), map1Keys.begin(), map1Keys.end());
    temp.insert(temp.end(), map2Keys.begin(), map2Keys.end());
}

```

```

sort(temp.begin(), temp.end());
temp.erase(unique(temp.begin(), temp.end()), temp.end());

auto it1 = map1Keys.begin();
auto it2 = map2Keys.begin();
stringstream ss1, ss2;
int count1, count2;

int time1, time2;
string internetTF1, internetTF2;
bool internet;
int size1, size2;
int lines1, lines2;
int index1, index2, index3;
string name1, name2;
string type1, type2;

string value1, value2;

for (size_t i = 0; i < temp.size(); i++)
{
    if (find(res.begin(), res.end(), i + 1) != res.end())
    {
        auto itIndex1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
        auto itIndex2 = find(map2Keys.begin(), map2Keys.end(), i + 1);

        index1 = distance(map1Keys.begin(), itIndex1);
        index2 = distance(map2Keys.begin(), itIndex2);

        ss1 = data1[index1]->getStr();
        ss2 = data2[index2]->getStr();

        value1 = ss1.str();
        value2 = ss2.str();

        count1 = count(value1.begin(), value1.end(), ' ');
        count2 = count(value2.begin(), value2.end(), ' ');

        ss1 >> name1;
        ss1 >> index1;
        ss1 >> time1;
        ss1 >> size1;
        ss1 >> lines1;
        ss1 >> internetTF1;
        if (count1 == 6)
            ss1 >> type1;

        ss2 >> name2;
        ss2 >> index2;
        ss2 >> time2;
        ss2 >> size2;
        ss2 >> lines2;
        ss2 >> internetTF2;
        if (count2 == 6)
            ss2 >> type2;

        name1 += name2;
        index1 += index2;
        time1 += time2;
        size1 += size2;
        lines1 += lines2;
        if (internetTF1 == "1" || internetTF2 == "1")
            internet = true;
        if (count1 == 6 || count2 == 6)
        {
            type1 += type2;
            resultMap.emplace(i + 1, new CMalware(internet, time1, size1, lines1, index1, name1,
type1));
        }
    }
}

```

```

        else
            resultMap.emplace(i + 1, new CProgram(internet, time1, size1, lines1, index1,
name1));
    }
    else
    {
        it1 = find(map1Keys.begin(), map1Keys.end(), i + 1);
        if (it1 != map1Keys.end())
        {
            index3 = distance(map1Keys.begin(), it1);
            resultMap.emplace(i + 1, data1[index3]);
        }
        else
        {
            it2 = find(map2Keys.begin(), map2Keys.end(), i + 1);
            index3 = distance(map2Keys.begin(), it2);
            resultMap.emplace(i + 1, data2[index3]);
        }
    }
}

map1Keys.erase(map1Keys.begin(), map1Keys.end());
map2Keys.erase(map2Keys.begin(), map2Keys.end());
temp.erase(temp.begin(), temp.end());
res.erase(res.begin(), res.end());
data1.erase(data1.begin(), data1.end());
data2.erase(data2.begin(), data2.end());

return resultMap;
}

```

Header.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE)

#include <string>
#include <iostream>
#include <iomanip>
#include <locale>
#include <fstream>
#include <sstream>
#include <istream>
#include <vector>
#include <memory>
#include <list>
#include <map>
#include <set>
#include <unordered_set>
#include <algorithm>
#include <iterator>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::boolalpha;
using std::setiosflags;
using std::ios;
using std::ifstream;
using std::ostream;
using std::ofstream;

```

```

using std::stringstream;
using std::istream;
using std::vector;
using std::list;
using std::map;
using std::set;
using std::unordered_set;
using std::unique_ptr;
using std::shared_ptr;
using std::advance;
using std::stoi;
using std::for_each;
using std::make_move_iterator;
using std::set_intersection;
using std::back_inserter;
using std::pair;
using std::transform;
using std::inserter;

```

malware.h

```

#pragma once
#include "program.h"
class CMalware final: public CProgram
{
private:
    string type;

public:
    string getInfo() const override final;
    stringstream getStr() const override final;
    bool elementOutput(int, string) override final;
    int countElement(int, string) override final;

    CMalware();
    CMalware(bool, int, int, int, int, string, string);
    CMalware(const CMalware&);
    ~CMalware() override final;

    bool operator==(const int) const override final;
};

```

program.h

```

#pragma once
#include "Header.h"

class CProgram
{ protected:
    int timeOfWork;           //average time of program execution
    int size;                 //size of program
    int amountOfLines;        //number of lines in code
    int index;                //index
    bool useInternet;          //use internet
    string name;              //name of program

public:
    virtual string getInfo() const;
    virtual stringstream getStr() const;
    int getID() const;
    virtual bool elementOutput(int, string);
    virtual int countElement(int, string);

    CProgram();
    CProgram(bool, int, int, int, int, string);
    CProgram(const CProgram&);
    virtual ~CProgram();

```

```

        friend ostream& operator<< (ostream&, const CProgram&);
        virtual bool operator==(const int) const;
};

```

Functor.cpp

```

#include "Functor.h"

bool Functor::operator() (const shared_ptr<CProgram>& program1, const shared_ptr<CProgram>&
program2)
{
    if (value % 2 != 0)
        return program1->getID() < program2->getID();
    else
        return program1->getID() > program2->getID();
}

Functor::Functor(int value) :value(value) {}
Functor::~~Functor() {}

```

Functor.h

```

#pragma once
#include "Program.h"

class Functor
{
private:
    int value;

public:
    bool operator()(const shared_ptr<CProgram>&, const shared_ptr<CProgram>&);
    Functor(int);
    ~Functor();
};

```


4. Результати роботи про грами

```
Выберите STL контейнер:
1. Vector
2. List
3. Map
4. Set
5. Выход
=====
Ваш выбор: 1

1)Вывод на экран
2)Удаление
3)Добавление
4)Объединить векторы
5)Сортировка
6)Завершение работы
=====
Ваш выбор: 1

Выберите команду:
1) Вывести весь список на экран
2) Вывести программу по ID
3) Вывести количество элементов по критерию
4) Найти элемент по критерию
5) Вернуться к выбору действий
=====
Ваш выбор: 1

      Название      Индекс  Время работы  Размер  Количество линий  Интернет  Тип
1. Basic           65      0           0        0             false
2. BestMalware     35634   8800        555       35             true      Exploit
3. Calculator      53453   423         523       654            false
4. MoneyStealer    67456   345         789       423            false      Rootkit

1)Вывод на экран
2)Удаление
3)Добавление
4)Объединить векторы
5)Сортировка
6)Завершение работы
=====
Ваш выбор: 4

Контейнер, с которым будет объединение:

      Название      Индекс  Время работы  Размер  Количество линий  Интернет  Тип
1. Browser         56789   534         164       4123           true
2. TextEditor      53425   423         6452      3122           true
3. JustMap         23567   231         534       634            false      Worm
4. Intired         90346   123         345       964            true       Adware

Векторы объединены.

1)Вывод на экран
2)Удаление
3)Добавление
4)Объединить векторы
5)Сортировка
6)Завершение работы
=====
Ваш выбор: 1

Выберите команду:
1) Вывести весь список на экран
2) Вывести программу по ID
3) Вывести количество элементов по критерию
4) Найти элемент по критерию
5) Вернуться к выбору действий
=====
Ваш выбор: 1

      Название      Индекс  Время работы  Размер  Количество линий  Интернет  Тип
1. Basic           65      0           0        0             false
2. BestMalware     35634   8800        555       35             true      Exploit
3. Calculator      53453   423         523       654            false
4. MoneyStealer    67456   345         789       423            false      Rootkit
5. Browser         56789   534         164       4123           true
6. TextEditor      53425   423         6452      3122           true
7. JustMap         23567   231         534       634            false      Worm
8. Intired         90346   123         345       964            true       Adware
```

```
Тест подсчёта элементов      выполнен успешно.
Тест сортировки вектора      выполнен успешно.

Векторы объединены.
Тест слияния векторов        выполнен успешно.

List
Тест добавления элемента     выполнен успешно.
Тест удаления элемента       выполнен успешно.
Тест получения элемента      выполнен успешно.
Тест подсчёта элементов      выполнен успешно.
Тест сортировки вектора      выполнен успешно.

Map
Тест добавления элемента     выполнен успешно.
Тест удаления элемента       выполнен успешно.
Тест получения элемента      выполнен успешно.
Тест подсчёта элементов      выполнен успешно.
Тест сортировки map          выполнен успешно.
Тест слияния map             выполнен успешно.

Set
Тест добавления элемента     выполнен успешно.
Тест удаления элемента       выполнен успешно.
Тест получения элемента      выполнен успешно.
Тест подсчёта элементов      выполнен успешно.
Тест сортировки сета         выполнен успешно.

Утечка памяти отсутствует.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з сортуванням STL контейнерів. У меню вектора та мапа було додано можливість конкатенації двох контейнерів та в усі меню було додано можливість сортування контейнерів.

Програма протестована, витоків пам'яті немає, виконується без помилок.