Практична робота №8

Тема: Використання контейнера std::list і асоціативний контейнера std::map.

Мета: Навчитись практично застосовувати контейнери std::list та std::map бібліотеки STL . Закріпити навичками використання ітераторів та алгоритмів для опрацювання вмісту контейнерів.

Хід роботи

1. У Git-репозиторію із попередніх практичних робіт створюю нову гілку «PR8» і переходжу в неї для виконання даної практичної роботи.

2. Створюю два контейнери std::list для зберігання цілих чисел. Перший заповнюю десятьма непарними непослідовними цілими числами, другий – з допомогою ітератора десятьма парними непослідовними значеннями.

3. Створюю третій контейнер std::list, в який з допомогою алгоритму std::merge() об’єдную два попередніх контейнери. Для виконання об’єднання з допомогою алгоритму std::merge() попередньо сортую списки. Для цього використовую алгоритм std::sort().

4. Виводжу вміст трьох списків std::list на екран.

5. Взявши за основу ієрархію класів, розроблену на попередніх практичних роботах згідно варіанту індивідуального завдання, в функції main створюю і заповнюю асоціативний контейнер std::map, в якому будуть зберігатись вказівники на базовий клас як значення, а ключем слугуватиме ідентифікатор об’єкта - ID.

6. Додаю в інтерактивне меню пункт для заповнення контейнера std::map об’єктами похідних класів.

7. Добадаю в інтерактивне меню пункти для виведення об’єкта з контейнера std::map згідно введеного значення ідентифікатора об’єкта (Id) для заповнення контейнера std::map об’єктами похідних класів.

8. Перевіряю роботу доданих пунктів меню - заповнюю std::map об’єктами похідних класів, після чого, за допомогою інтерактивного меню, виводжу пару об’єктів, зазначивши потрібний id.

9. Роблю коміт проєкту із повідомленням «done practical work №8».

10. Об’єдную гілку PR8 у гілку main (merge) і надсилаю зміни у гілці main у віддалений репозиторій (git push).

11. Оформлюю звіт.

Лістинг розробленого коду

Main.cpp

```
#include <iostream>
```

```
#include "Car.h"
#include "Bus.h"
#include "Vehicle.h"
#include "CustomVector.h"
#include <vector>
#include <algorithm>
#include <list>
#include <map>

using namespace std;

int randInt(int minInclusiveValue, int maxInclusiveValue) {
    return (minInclusiveValue + (rand() % maxInclusiveValue - minInclusiveValue + 1));
}

int randIntOdd(int minInclusiveValue, int maxInclusiveValue) {
    int randOdd = 0;

    do {
        randOdd = randInt(minInclusiveValue, maxInclusiveValue);
    } while (randOdd % 2 == 0);

    return randOdd;
}

int randIntEven(int minInclusiveValue, int maxInclusiveValue) {
    int randOdd = 0;

    do {
        randOdd = randInt(minInclusiveValue, maxInclusiveValue);
    } while (randOdd % 2 != 0);

    return randOdd;
}

int main()
{
    srand(time(nullptr));

#pragma region lab8

    //Task 2

    list<int> ints1(10);
    list<int> ints2(10);

    for (auto iterator = ints1.begin(); iterator != ints1.end(); iterator++) {
        *iterator = randIntOdd(1, 100);
    }

    for (auto iterator = ints2.begin(); iterator != ints2.end(); iterator++) {
        *iterator = randIntEven(1, 100);
    }

    for (int element : ints1) {
        cout << element << ", ";
    }

    cout << endl << "list 1" << endl;

    for (int element : ints2) {
```

```
        cout << element << ", ";
    }

    cout << endl << "list 2" << endl;

    //task 3

    ints1.sort();
    ints2.sort();
    list<int> ints3;
    ints3.merge(ints1);
    ints3.merge(ints2);

    //task 4

    for (int element : ints1) {
        cout << element << ", ";
    }

    cout << endl << "Output first list" << endl;

    for (int element : ints2) {
        cout << element << ", ";
    }

    cout << endl << "Output second list" << endl;

    for (int element : ints3) {
        cout << element << ", ";
    }

    cout << endl << "Output third list" << endl;

    //task 5

    map<int, Vehicle*> vehiclesMap;

    int choose, id;

    while (true) {
        cout << "1. Create Car \n2. Create Bus \n3. Print vehicle information \n0. Exit \nWhat
would you want to do?: "; cin >> choose;

        if (choose == 3) {
            if (!vehiclesMap.empty()) {
                cout << "Input id: "; cin >> id;

                try {
                    cout << endl;
                    auto it = vehiclesMap.find(id);
                    it->second->output();
                }
                catch (string a) {
                    cout << "Given id caused the exception: " << a;
                }

            }
            else {
                cout << "The map is empty!";
            }
        }
```

```
        if (choose == 1) {
            int objectId;
            cout << "Input object id: "; cin >> objectId;
            Car* car = new Car;
            car->input();
            vehiclesMap.insert({ objectId, car });
        }

        else if (choose == 2) {
            int objectId;
            cout << "Input object id: "; cin >> objectId;
            Bus* bus = new Bus;
            bus->input();
            vehiclesMap.insert({ objectId, bus });
        }

        else if (choose == 0) {
            exit(0);
        }

        cout << endl << endl << endl;
    }

#pragma endregion (done)
}
```

Висновок: Під час виконання даної практичної роботи, здобуто навички практичного застосовувати контейнерів std::list та std::map бібліотеки STL . Закріплено навички використання ітераторів та алгоритмів для опрацювання вмісту контейнерів.