Міністерство освіти і науки України
КПІ ім. Ігоря Сікорського
Кафедра ІПІ

ЗВІТ

з виконання лабораторної роботи № 6
з кредитного модуля
“Основи програмування-2. Методології програмування”
Варіант № 22

Виконав:
студент 1-го курсу
гр. ІП-22 ФІОТ
Підпанюк Віталій Андрійович

Київ 2023

Завдання:

22. Спроектувати АТД "Мультимножина" для контейнера, що містить дані довільного типу. Інтерфейс АТД включає такі обов'язкові операції:

- визначення пустоти множини,
- перевірка приналежності елемента множині,
- видалення елемента з множини,
- перетин двох множин,
- ітератор для доступу до елементів списку з операціями:
  1) встановлення на початок множини,
  2) перевірка кінця множини,
  3) доступ до значення поточного елемента множини,
  4) перехід до наступного елемента множини.

Код програми:

main.cpp:

```
#include <iostream>
#include "function.h"
using namespace std;


int main() {
    type_choose();
    return 0;
}
```

multiset.h:

```
#pragma once
#include <iostream>
//#include <set>

using namespace std;

template<typename T>
class MultiSet {
private:

        struct Node {
        T value;
        Node *next;
```

```
        Node(const T &val) : value(val), next(nullptr) {}
    };

    Node *head;

public:
    MultiSet();

    void insert(const T &element);

    bool contains(const T &element);

    void merge(const MultiSet<T> &otherSet);

    void clear();

    class Iterator {
    private:
        const MultiSet<T> &multiSet;
        Node *current;

    public:
        Iterator(const MultiSet<T> &set);

        void begin();

        bool end();

        const T &value();

        void next();
    };

Iterator getIterator() const;
};
```

multiset.tpp:

```
#pragma once
#include <iostream>
#include "multiset.h"
using namespace std;
```

```
template<typename T>
MultiSet<T>::MultiSet() : head(nullptr) {}

template<typename T>
void MultiSet<T>::insert(const T& element) {
     Node* newNode = new Node(element);

    if (head == nullptr)
        head = newNode;
    else if (element <= head->value) {
        newNode->next = head;
        head = newNode;
    }
    else {
        Node* current = head;
        while (current->next != nullptr && element >
current->next->value)
            current = current->next;

        newNode->next = current->next;
        current->next = newNode;
    }
}

template<typename T>
bool MultiSet<T>::contains(const T& element) {
    Node* current = head;
    while (current != nullptr) {
        if (current->value == element)
            return true;
        current = current->next;
    }
    return false;
}

template<typename T>
void MultiSet<T>::merge(const MultiSet<T>& otherSet) {
    Node* current = otherSet.head;
    while (current != nullptr) {
        insert(current->value);
```

```
        current = current->next;
    }
}

template<typename T>
void MultiSet<T>::clear() {
    Node* current = head;
    while (current != nullptr) {
        Node* next = current->next;
        delete current;
        current = next;
    }
    head = nullptr;
}

template<typename T>
MultiSet<T>::Iterator::Iterator(const MultiSet<T>& set) :
multiSet(set), current(set.head) {}

template<typename T>
void MultiSet<T>::Iterator::begin() {
    current = multiSet.head;
}

template<typename T>
bool MultiSet<T>::Iterator::end() {
    return current == nullptr;
}

template<typename T>
const T& MultiSet<T>::Iterator::value() {
    return current->value;
}

template<typename T>
void MultiSet<T>::Iterator::next() {
    current = current->next;
}

template<typename T>
```

```
typename MultiSet<T>::Iterator MultiSet<T>::getIterator()
const {
    return Iterator(*this);
}
```

functions.cpp:

```
#include "function.h"
//#include "multiset.h"

void type_choose() {
    int chosen_type = 0;

    cout << "Enter 1 for creating a multiset of int
numbers, 2 for double, or 3 for char: ";
    cin >> chosen_type;

    if (chosen_type == 1) {

        MultiSet<int> mySet;
        program(mySet);

    }

    else if (chosen_type == 2) {

        MultiSet<double> mySet;
        program(mySet);

    }

    else {

        MultiSet<char> mySet;
        program(mySet);

    }

}

template <typename T>
void program(MultiSet<T>& mySet)
```

```
{
    T insert;
    MultiSet<T> newSet;
    int start_quantity;
    cout<<"Enter start quantity of elements:\n";
    cin>>start_quantity;
    cout<<"Enter elements:";
    for(int i = 0; i < start_quantity; i++)
    {
        cin>>insert;
        mySet.insert(insert);
    }
    cout<<"Add new element:\n";
    cin>>insert;
    mySet.insert(insert);
    cout<<"Create new multiset to merge old and new\n";
    cout<<"Enter start quantity of elements:\n";
    cin>>start_quantity;
    cout<<"Enter elements:";
    for(int i = 0; i < start_quantity; i++)
    {
        cin>>insert;
        newSet.insert(insert);
    }
    mySet.merge(newSet);
    cout<<"Merge success\n";
    newSet.clear();

    typename MultiSet<T>::Iterator iterator =
mySet.getIterator();
    iterator.begin();

    cout << "-------------------------------------" <<
endl;

    while (!iterator.end()) {
        cout << iterator.value() << " ";
        iterator.next();
    }
```

```
    cout << endl << "-----------------------------------
" << endl;
    mySet.clear();
}
```

functions.h:

```
#pragma once
#include <iostream>
#include "multiset.tpp"
using namespace std;

template <typename T>
void program(MultiSet<T>& mySet);
void type_choose();
```

Тестування програми:

```
C:\my__folder\study\OOP\lab_6\project_1\cmake-build-debug\project_1.exe
Enter 1 for creating a multiset of int numbers, 2 for double, or 3 for char: 1
 Enter start quantity of elements:
5
Enter elements:2 3 4 5 6
Add new element:
4
Create new multiset to merge old and new
Enter start quantity of elements:
2
Enter elements:8 9
Merge success
-----------------------------------
2 3 4 4 5 6 8 9
-----------------------------------
```