Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №7

з дисципліни «Об’єктно-орієнтоване програмування»

На тему «ВВЕДЕННЯ В УЗАГАЛЬНЕНЕ ПРОГРАМУВАННЯ»

Варіант №20

Виконав:

Студент групи КНТ-122	О. А. Онищенко

Прийняли:

Ст. Викладач	Т. В. Голуб
Ст. Викладач	Л. Ю. Дейнега

2023

Мета роботи

Навчитись використовувати шаблони при створені програм мовою програмування С++.

Текст завдання №5

1. Виконати завдання з лабораторної роботи №1, де тип елементу заданої структури даних довільний. Використати шаблонні функції.
2. Виконати завдання з лабораторної роботи №5 з довільним типом даних. Використати шаблонні класи відповідно до завдання викладача.
3. Виконати завдання з лабораторної роботи №1 використавши, для зберігання даних класи Standard Template Library (STL) або list або vector. Поясніть різницю в використанні цих класів.

1) Код програми – main.cpp

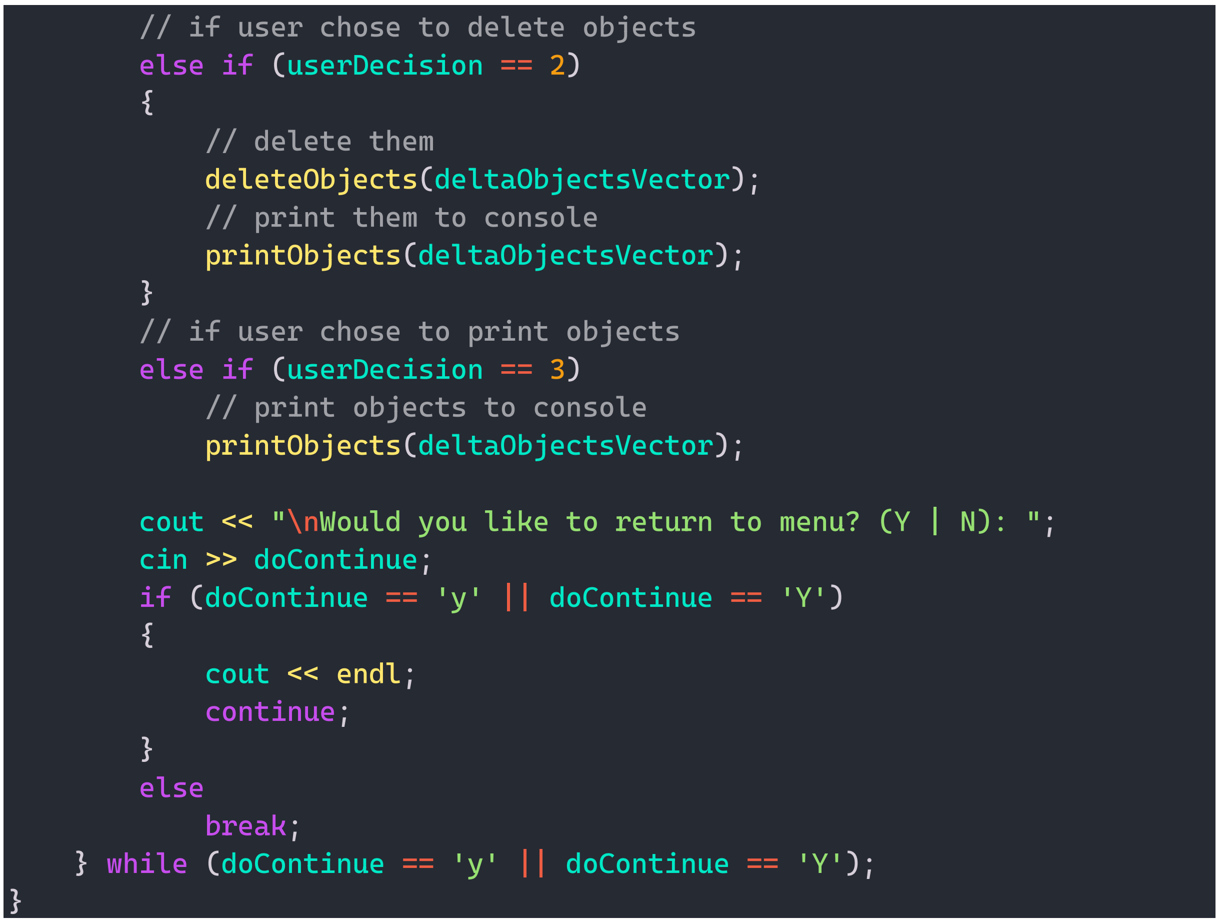
```cpp
// include necessary libraries
#include "D:\repos\university\lib.h"
#include "sup.h"

// Створити клас Delta таким чином, щоб кожний об’єкт вміщував свій
персональний номер (дескриптор об’єкта) та функцію, яка повертає його
значення. Дескриптор об’єкта – унікальне для об’єктів даного типу ціле
число.
// Виконати завдання з лабораторної роботи №1, де тип елементу заданої
структури даних довільний. Використати шаблонні функції.

// func main start
int main()
{
    // declare local variables //
    srand(time(NULL));
    char doContinue;
    ll userDecision;
    /////////////////////////////

    // project intro
    cout << "\n";
    do
    {
        ///////////////////////////////////////

        char doReturnToMenu;
        // output menu to user and prompt them to choose an option
        cout << BOLD << "Choose a data type\n"
             << UNBOLD;
        cout << "1. Integer\n";
        cout << "2. Double\n";
        cout << "3. String\n";
        cout << "4. Exit\n";
        cout << "Enter: ";
        userDecision = getNum();
        cout << endl;

        // if user chose to add objects
        if (userDecision == 1)
        {
            vector<unique_ptr<Delta<int>>> deltaObjectsVector;
            outputMenu(deltaObjectsVector);
        }
        // if user chose to delete objects
        else if (userDecision == 2)
        {
            vector<unique_ptr<Delta<double>>> deltaObjectsVector;
            outputMenu(deltaObjectsVector);
        }
        // if user chose to print objects
        else if (userDecision == 3)
        {
            vector<unique_ptr<Delta<string>>> deltaObjectsVector;
            outputMenu(deltaObjectsVector);
        }

        ///////////////////////////////////////

        // ask user if they would like to continue execution of program
        cout << "\nWould you like to continue program execution? (Y | N):
";
        cin >> doContinue;
        if (doContinue == 'Y' || doContinue == 'y')
        {
            cout << "\n\n";
            continue;
        }
        else
            break;
    } while (doContinue = 'Y' || doContinue == 'y');

    // stop main function execution
    cout << "\nThanks for using this program\n\n";
    return 0;
}
```

1) Код програми – sup.h

```
#include "sup.h"
#include "D:\repos\university\lib.h"

// Створити клас Delta таким чином, щоб кожний об’єкт вміщував свій
персональний номер (дескриптор об’єкта) та функцію, яка повертає його
значення. Дескриптор об’єкта – унікальне для об’єктів даного типу ціле
число.
// Виконати завдання з лабораторної роботи №1, де тип елементу заданої
структури даних довільний. Використати шаблонні функції.

// delta class declaration
template <typename T>
class Delta
{
private:
    // declare private members
    ll descriptor;
    T value;

public:
    // default constructor for when we don't specify the value
    Delta() : value(), descriptor(nextDescriptor()) {}
    // parameterised constructor for when we do specify the value
    Delta(T inValue) : value(inValue), descriptor(nextDescriptor()) {}

    static ll nextDescriptor()
    {
        static ll nextDescriptor = 0;
        return nextDescriptor++;
    }

    // next identifier creator
    T getValue() const
```

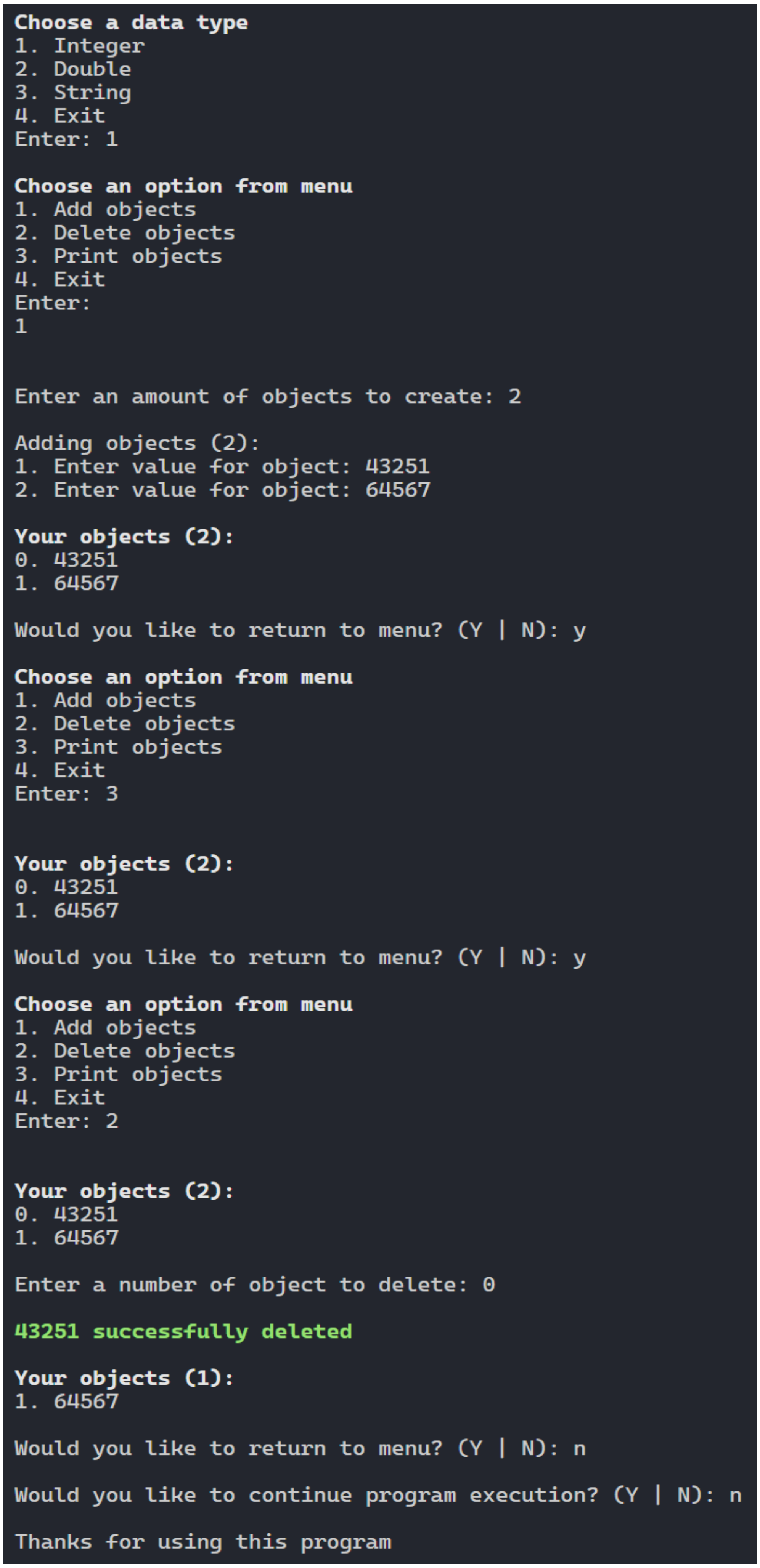
```cpp
    {
        return value;
    }

    ll getDescriptor() const
    {
        return descriptor;
    }

    void setValue(T inValue)
    {
        value = inValue;
    }

    // create default destructor function
    ~Delta()
    {
    }
};

// object creation function
template <typename T>
void createObjects(vector<unique_ptr<Delta<T>>> &deltaObjectsVector)
{
    // ask user to enter number of delta objects to create
    cout << "\nEnter an amount of objects to create: ";
    ll objectsAmount = getNum();

    // if entered amount is less than one
    if (objectsAmount < 1)
    {
        // output error and stop function
        cout << RED << "\nERROR: Invalid amount of objects...\n\n"
             << UNRED;
        // stop function execution
        return;
    }

    // create specified amount of objects using a for loop
    cout << "\nAdding objects (" << objectsAmount << "):\n";
    for (ll i = 0; i < objectsAmount; i++)
    {
        cin.ignore();
        T value;
        cout << i + 1 << ". Enter value for object: ";
        cin >> value;
        deltaObjectsVector.push_back(make_unique<Delta<T>>(value));
    }
    // end function execution
    return;
}

// printing objects function
template <typename T>
void printObjects(vector<unique_ptr<Delta<T>>> &deltaObjectsVector)
{
    // get vector size
    ll vectorSize = deltaObjectsVector.size();
    if (vectorSize == 0)
    {
        bad("No objects to print");
        return;
    }

    // output all objects with their identifiers using a for loop
    cout << BOLD << "\nYour objects (" << vectorSize << "): \n"
         << UNBOLD;
    for (ll i = 0; i < vectorSize; i++)
    {
        cout << deltaObjectsVector[i]->getDescriptor() << ". " <<
deltaObjectsVector[i]->getValue() << endl;
    }
}

// object deletion function
template <typename T>
void deleteObjects(vector<unique_ptr<Delta<T>>> &deltaObjectsVector)
{
    // check if vector is empty
    if (deltaObjectsVector.size() == 0)
        // if so, output error message
        cout << GRAY << "\nNo objects to delete\n"
             << UNGRAY;
    // if not
    else
    {
        // print all objects to user
        printObjects(deltaObjectsVector);

        cout << "\nEnter a number of object to delete: ";
        ll numToDelete = getNum();

        // if the object number is out of range
        if (numToDelete > deltaObjectsVector.size() - 1 || numToDelete <
0)
```

```cpp
            // output error message
            cout << RED << "\nERROR: Invalid object number\n"
                 << UNRED;
        // if not
        else
        {
            // output success message
            cout << GREEN << endl
                 << deltaObjectsVector[numToDelete]->getValue() << " successfully deleted\n"
                 << UNGREEN;

            // erase object from vector
            deltaObjectsVector.erase(deltaObjectsVector.begin() + numToDelete);
        }
    }

    // end function execution
    return;
}

// for showing the main menu of the application
template <typename T>
void outputMenu(vector<unique_ptr<Delta<T>>> &deltaObjectsVector)
{
    char doContinue;
    do
    {
        // output menu to user and prompt them to choose an option
        cout << BOLD << "Choose an option from menu\n"
             << UNBOLD;
        cout << "1. Add objects\n";
        cout << "2. Delete objects\n";
        cout << "3. Print objects\n";
        cout << "4. Exit\n";
        cout << "Enter: ";
        ll userDecision = getNum();
        cout << endl;

        // if user chose to add objects
        if (userDecision == 1)
        {
            // add objects
            createObjects(deltaObjectsVector);
            // print them to console
            printObjects(deltaObjectsVector);
        }
```



```cpp
        // if user chose to delete objects
        else if (userDecision == 2)
        {
            // delete them
            deleteObjects(deltaObjectsVector);
            // print them to console
            printObjects(deltaObjectsVector);
        }
        // if user chose to print objects
        else if (userDecision == 3)
            // print objects to console
            printObjects(deltaObjectsVector);

        cout << "\nWould you like to return to menu? (Y | N): ";
        cin >> doContinue;
        if (doContinue == 'y' || doContinue == 'Y')
        {
            cout << endl;
            continue;
        }
        else
            break;
    } while (doContinue == 'y' || doContinue == 'Y');
}
```

1) Приклад роботи



```
Choose a data type
1. Integer
2. Double
3. String
4. Exit
Enter: 1

Choose an option from menu
1. Add objects
2. Delete objects
3. Print objects
4. Exit
Enter:
1


Enter an amount of objects to create: 2

Adding objects (2):
1. Enter value for object: 43251
2. Enter value for object: 64567

Your objects (2):
0. 43251
1. 64567

Would you like to return to menu? (Y | N): y

Choose an option from menu
1. Add objects
2. Delete objects
3. Print objects
4. Exit
Enter: 3


Your objects (2):
0. 43251
1. 64567

Would you like to return to menu? (Y | N): y

Choose an option from menu
1. Add objects
2. Delete objects
3. Print objects
4. Exit
Enter: 2


Your objects (2):
0. 43251
1. 64567

Enter a number of object to delete: 0

43251 successfully deleted

Your objects (1):
1. 64567

Would you like to return to menu? (Y | N): n

Would you like to continue program execution? (Y | N): n

Thanks for using this program
```

2) Код програми – main.cpp

```
#include "D:\repos\university\lib.h"
#include "sup.h"
using namespace std;

// Виконати завдання з лабораторної роботи #5 з довільним типом даних

int main()
{
    srand(time(NULL));
    char doContinue;
    vector<unique_ptr<DynamicString>> container;
    char doReturnToMenu;

    cout << "\n";
    do
    {
        ///////////////////////////////////////

        do
        {
            outputMenu(container);

            cout << "\nWould you like to return to menu? (Y | N): ";
            cin >> doReturnToMenu;
            if (doReturnToMenu == 'Y' || doReturnToMenu == 'y')
            {
                cout << endl
                     << endl;
                continue;
            }
            else
                break;

        } while (doReturnToMenu == 'y' || doReturnToMenu == 'Y');

        ///////////////////////////////////////

        cout << "\nWould you like to continue program execution? (Y | N): 
";
        cin >> doContinue;
        if (doContinue == 'Y' || doContinue == 'y')
        {
            cout << "\n\n";
            continue;
        }
        else
            break;
    } while (doContinue = 'Y' || doContinue == 'y');

    cout << "\nThanks for using this program\n\n";
    return 0;
}
```

2) Код програми – sup.h

```
#include "sup.h"
#include "D:\repos\university\lib.h"

// Виконати завдання з лабораторної роботи #5 з довільним типом даних

const string ROOT_PATH = "D:/repos/university/year1-term2/OOP/lb7/tsk2/";

class DynamicString
{
private:
    string m_value;
    ll m_size;

public:
    DynamicString() : m_value(""), m_size(0) {}
    DynamicString(string value) : m_value(value), m_size(value.length())
{}
    void setValue(string value)
    {
        m_value = value;
        m_size = value.length();
    }
    string getValue() const { return m_value; }
    ll getSize() const { return m_size; }
    void m_reverse() { reverse(m_value.begin(), m_value.end()); }
    void m_replace(string replaceThis, string withThis) {
m_value.replace(m_value.find(replaceThis), replaceThis.length(),
withThis); }
    void m_remove_spaces()
    {
        stringstream ss(m_value);
        string word;
        m_value.clear();
```

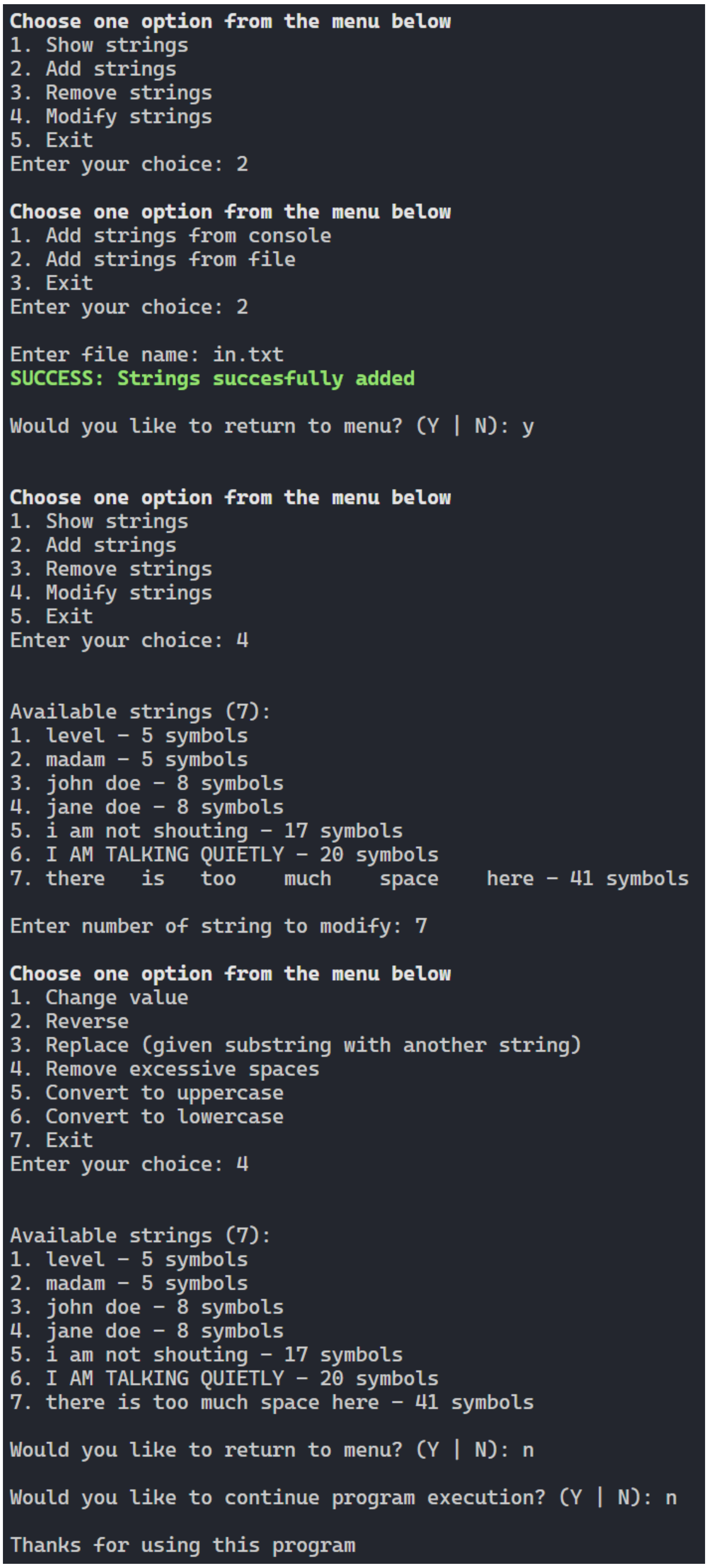
```cpp
        while (ss >> word)
            m_value += word + " ";

        if (!m_value.empty())
            m_value.pop_back();
    }
    void m_to_upper() { transform(m_value.begin(), m_value.end(),
m_value.begin(), ::toupper); }
    void m_to_lower() { transform(m_value.begin(), m_value.end(),
m_value.begin(), ::tolower); }
    ~DynamicString() {}
};

void showStrings(vector<unique_ptr<DynamicString>> &container)
{
    ll stringsNum = container.size();
    if (stringsNum == 0)
    {
        bad("No strings found");
        return;
    }

    cout << "Available strings (" << stringsNum << "):\n";
    for (ll i = 0; i < stringsNum; i++)
    {
        cout << i + 1 << ". " << container[i]->getValue() << " - " <<
container[i]->getSize() << " symbols\n";
    }
}

void showStrings(vector<unique_ptr<DynamicString>> &container, const
string &FILE)
{
    ll stringsNum = container.size();
    if (stringsNum == 0)
    {
        bad("No strings found");
        return;
    }

    ofstream file(FILE);

    if (!file.is_open())
    {
        bad("File could not be opened");
        return;
    }
    file << "==============================\n\n";
    file << "Available strings (" << stringsNum << "):\n";
    for (ll i = 0; i < stringsNum; i++)
    {
        file << i + 1 << ". " << container[i]->getValue() << " - " <<
container[i]->getSize() << " symbols\n";
    }
    file << "\n==============================\n\n";
    file.close();

    if (file.good())
        good("Strings succesfully saved");
    else
        bad("Strings were not saved");
}

void addStrings(vector<unique_ptr<DynamicString>> &container)
{
    ll initSize = container.size();

    cout << "Enter number of strings to add: ";
    ll numToAdd = getNum();
    cout << endl;
    cin.ignore();

    if (numToAdd == 0)
    {
        bad("Enter a valid number of strings");
        return;
    }

    for (ll i = 0; i < numToAdd; i++)
    {
        string value;
        cout << i + 1 << ". Enter value: ";
        getline(cin, value);
        container.push_back(make_unique<DynamicString>(value));
    }
    cout << endl;

    if (container.size() == initSize + numToAdd)
        good("Strings succesfully added");
    else
        bad("Strings were not added");

    cout << endl;
    showStrings(container);
}
```

```cpp
void addStrings(vector<unique_ptr<DynamicString>> &container, const
string &FILE)
{
    ll initSize = container.size();
    ifstream file(FILE);
    string line;
    vector<string> lines;

    if (!file.is_open())
    {
        bad("File not found");
        return;
    }

    while (getline(file, line))
    {
        lines.push_back(line);
    }
    file.close();

    for (auto &line : lines)
    {
        container.push_back(make_unique<DynamicString>(line));
    }

    if (container.size() == initSize + lines.size())
        good("Strings succesfully added");
    else
        bad("Strings were not added");
}

void removeString(vector<unique_ptr<DynamicString>> &container)
{
    ll initSize = container.size();

    if (initSize == 0)
    {
        bad("No strings found");
        return;
    }

    cout << endl;
    showStrings(container);
    cout << endl;

    cout << "Enter number of string to remove: ";
    ll numToRemove = getNum();
    numToRemove--;

    if (numToRemove < 0 || numToRemove >= initSize)
    {
        bad("Enter a valid number string number");
        return;
    }

    container.erase(container.begin() + numToRemove);
    cout << endl;

    if (container.size() == initSize - 1)
        good("String succesfully removed");
    else
        bad("String was not removed");

    cout << endl;
    showStrings(container);
}

void modifyString(vector<unique_ptr<DynamicString>> &container)
{
    if (container.size() == 0)
    {
        bad("No strings found");
        return;
    }

    cout << endl;
    showStrings(container);
    cout << endl;

    cout << "Enter number of string to modify: ";
    ll numToModify = getNum();
    numToModify--;
    cout << endl;

    if (numToModify < 0 || numToModify >= container.size())
    {
        bad("Enter a valid number string number");
        return;
    }

    vector<string> menuItems = {
        "Change value",
        "Reverse",
        "Replace (given substring with another string)",
        "Remove excessive spaces",
        "Convert to uppercase",
        "Convert to lowercase",
        "Exit"};
    ll userDecision = showMenu(menuItems);
    cin.ignore();

    if (userDecision == 1)
    {
        string value;
        cout << "Enter new value: ";
        getline(cin, value);
        container[numToModify]->setValue(value);
    }
    else if (userDecision == 2)
    {
        container[numToModify]->m_reverse();
    }
    else if (userDecision == 3)
    {
        string replaceThis;
        string withThis;
        cout << "Enter substring to replace: ";
        getline(cin, replaceThis);
        cout << "Enter substring to replace with: ";
        getline(cin, withThis);
        container[numToModify]->m_replace(replaceThis, withThis);
    }
    else if (userDecision == 4)
    {
        container[numToModify]->m_remove_spaces();
    }
    else if (userDecision == 5)
    {
        container[numToModify]->m_to_upper();
    }
    else if (userDecision == 6)
    {
        container[numToModify]->m_to_lower();
    }

    cout << endl;
    showStrings(container);
}

void outputMenu(vector<unique_ptr<DynamicString>> &container)
{
    vector<string> menuItems = {
        "Show strings",
```

```cpp
        "Add strings",
        "Remove strings",
        "Modify strings",
        "Exit"};
    ll userDecision = showMenu(menuItems);

    if (userDecision == 1)
    {
        menuItems = {
            "Show in console",
            "Show in file",
            "Exit"};
        userDecision = showMenu(menuItems);

        if (userDecision == 1)
            showStrings(container);
        else if (userDecision == 2)
        {
            string fileName = ROOT_PATH;
            fileName += getFileName();
            showStrings(container, fileName);
        }
    }
    else if (userDecision == 2)
    {
        menuItems = {
            "Add strings from console",
            "Add strings from file",
            "Exit"};
        userDecision = showMenu(menuItems);

        if (userDecision == 1)
            addStrings(container);
        else if (userDecision == 2)
        {
            string fileName = ROOT_PATH;
            fileName += getFileName();
            addStrings(container, fileName);
        }
    }
    else if (userDecision == 3)
    {
        removeString(container);
    }
    else if (userDecision == 4)
    {
        modifyString(container);
    }
```

```
}
```

2) Приклад роботи



```
Choose one option from the menu below
1. Show strings
2. Add strings
3. Remove strings
4. Modify strings
5. Exit
Enter your choice: 2

Choose one option from the menu below
1. Add strings from console
2. Add strings from file
3. Exit
Enter your choice: 2

Enter file name: in.txt
SUCCESS: Strings succesfully added

Would you like to return to menu? (Y | N): y


Choose one option from the menu below
1. Show strings
2. Add strings
3. Remove strings
4. Modify strings
5. Exit
Enter your choice: 4


Available strings (7):
1. level - 5 symbols
2. madam - 5 symbols
3. john doe - 8 symbols
4. jane doe - 8 symbols
5. i am not shouting - 17 symbols
6. I AM TALKING QUIETLY - 20 symbols
7. there   is   too    much    space    here - 41 symbols

Enter number of string to modify: 7

Choose one option from the menu below
1. Change value
2. Reverse
3. Replace (given substring with another string)
4. Remove excessive spaces
5. Convert to uppercase
6. Convert to lowercase
7. Exit
Enter your choice: 4


Available strings (7):
1. level - 5 symbols
2. madam - 5 symbols
3. john doe - 8 symbols
4. jane doe - 8 symbols
5. i am not shouting - 17 symbols
6. I AM TALKING QUIETLY - 20 symbols
7. there is too much space here - 41 symbols

Would you like to return to menu? (Y | N): n

Would you like to continue program execution? (Y | N): n

Thanks for using this program
```

3) Код програми – main.cpp

```cpp
#include "D:\repos\university\lib.h"
#include "sup.h"

// Виконати завдання з лабораторної роботи #1 використавши, для
зберігання даних класи Standard Template Library (STL) або list або
vector. Поясніть різницю в використанні цих класів.

int main()
{
    srand(time(NULL));
    char doContinue;
    char doReturnToMenu;

    vector<unique_ptr<Delta>> container;
    list<unique_ptr<Delta>> container_list;

    cout << "\n";
    vector<string> menuItems = {
        "Vector",
        "List",
    };
    ll userDecision = showMenu(menuItems);
    do
    {
        do
        {
            ///////////////////////////////////////

            if (userDecision == 1)
            {
                outputMenu(container);
            }
            else if (userDecision == 2)
            {
                outputMenu(container_list);
            }

            ///////////////////////////////////////

            cout << "\nWould you like to return to menu? (Y | N): ";
            cin >> doReturnToMenu;
            if (doReturnToMenu == 'Y' || doReturnToMenu == 'y')
            {
                cout << endl
                     << endl;
                continue;
            }
            else
                break;

        } while (doReturnToMenu == 'y' || doReturnToMenu == 'Y');

        cout << "\nWould you like to continue program execution? (Y | N):
";
        cin >> doContinue;
        if (doContinue == 'Y' || doContinue == 'y')
        {
            cout << "\n\n";
            continue;
        }
        else
            break;
    } while (doContinue = 'Y' || doContinue == 'y');

    cout << "\nThanks for using this program\n\n";
    return 0;
}
```

3) Код програми – sup.h

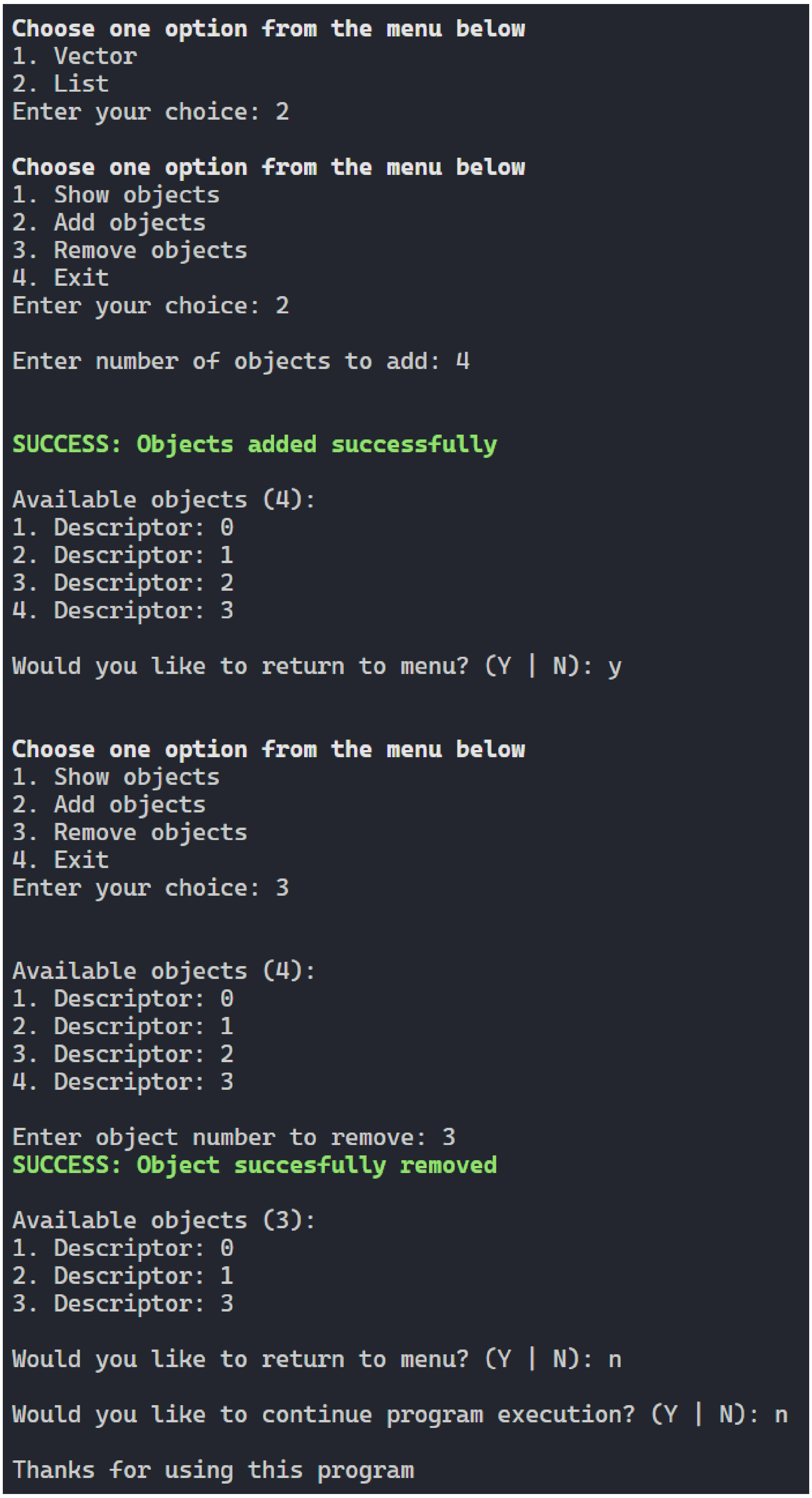
```
#include "D:\repos\university\lib.h"
#include "sup.h"

// Виконати завдання з лабораторної роботи #1 використавши, для
зберігання даних класи Standard Template Library (STL) або list або
vector. Поясніть різницю в використанні цих класів.

class Delta
{
private:
    ll m_id;

public:
    Delta() : m_id(nextID()) {}
    ll nextID()
    {
        static ll id = 0;
        return id++;
    }
```

```cpp
    ll getID() const { return m_id; }
    ~Delta() {}
};

void showObjs(vector<unique_ptr<Delta>> &container)
{
    ll objsNum = container.size();
    if (objsNum == 0)
    {
        bad("No objects to show");
        return;
    }

    cout << "Available objects (" << objsNum << "):\n";
    for (ll i = 0; i < objsNum; i++)
    {
        cout << i + 1 << ". Descriptor: " << container[i]->getID() <<
endl;
    }
}

void showObjs(list<unique_ptr<Delta>> &container)
{
    ll objsNum = container.size();
    if (objsNum == 0)
    {
        bad("No objects to show");
        return;
    }

    cout << "Available objects (" << objsNum << "):\n";
    ll i = 0;
    for (auto it = container.begin(); it != container.end(); it++)
    {
        cout << i + 1 << ". Descriptor: " << it->get()->getID() << endl;
        i++;
    }
}

void addObjs(vector<unique_ptr<Delta>> &container)
{
    ll initSize = container.size();

    cout << "Enter number of objects to add: ";
    ll numToAdd = getNum();
    cout << endl;

    if (numToAdd == 0)
    {
        bad("Enter a valid number of objects");
        return;
    }

    for (ll i = 0; i < numToAdd; i++)
    {
        container.push_back(make_unique<Delta>());
    }
    cout << endl;

    if (container.size() == initSize + numToAdd)
        good("Objects added successfully");
    else
        bad("Failed to add objects");

    cout << endl;
    showObjs(container);
}

void addObjs(list<unique_ptr<Delta>> &container)
{
    ll initSize = container.size();

    cout << "Enter number of objects to add: ";
    ll numToAdd = getNum();
    cout << endl;

    if (numToAdd == 0)
    {
        bad("Enter a valid number of objects");
        return;
    }

    for (ll i = 0; i < numToAdd; i++)
    {
        container.push_back(make_unique<Delta>());
    }
    cout << endl;

    if (container.size() == initSize + numToAdd)
        good("Objects added successfully");
    else
        bad("Failed to add objects");

    cout << endl;
    showObjs(container);
}
```

```cpp
void delObjs(vector<unique_ptr<Delta>> &container)
{
    ll initSize = container.size();

    if (initSize == 0)
    {
        bad("No objects found");
        return;
    }

    cout << endl;
    showObjs(container);
    cout << endl;

    cout << "Enter object number to remove: ";
    ll numToRemove = getNum();
    numToRemove--;

    if (numToRemove < 0 || numToRemove >= initSize)
    {
        bad("Enter a valid object number");
        return;
    }

    container.erase(container.begin() + numToRemove);
    cout << endl;

    if (container.size() == initSize - 1)
        good("Object succesfully removed");
    else
        bad("Object was not removed");

    cout << endl;
    showObjs(container);
}

void delObjs(list<unique_ptr<Delta>> &container)
{
    ll initSize = container.size();

    if (initSize == 0)
    {
        bad("No objects found");
        return;
    }

    cout << endl;
    showObjs(container);
    cout << endl;

    cout << "Enter object number to remove: ";
    ll numToRemove = getNum();
    numToRemove--;

    if (numToRemove < 0 || numToRemove >= initSize)
    {
        bad("Enter a valid object number");
        return;
    }

    auto it = next(container.begin(), numToRemove);
    container.erase(it);

    if (container.size() == initSize - 1)
        good("Object succesfully removed");
    else
        bad("Object was not removed");

    cout << endl;
    showObjs(container);
}

void outputMenu(vector<unique_ptr<Delta>> &container)
{
    vector<string> menuItems = {
        "Show objects",
        "Add objects",
        "Remove objects",
        "Exit"};
    ll userDecision = showMenu(menuItems);

    if (userDecision == 1)
    {
        showObjs(container);
    }
    else if (userDecision == 2)
    {
        addObjs(container);
    }
    else if (userDecision == 3)
    {
        delObjs(container);
    }
}
```

```
void outputMenu(list<unique_ptr<Delta>> &container)
{
    vector<string> menuItems = {
        "Show objects",
        "Add objects",
        "Remove objects",
        "Exit"};
    ll userDecision = showMenu(menuItems);

    if (userDecision == 1)
    {
        showObjs(container);
    }
    else if (userDecision == 2)
    {
        addObjs(container);
    }
    else if (userDecision == 3)
    {
        delObjs(container);
    }
}
```

3) Приклад роботи


```
Choose one option from the menu below
1. Vector
2. List
Enter your choice: 2

Choose one option from the menu below
1. Show objects
2. Add objects
3. Remove objects
4. Exit
Enter your choice: 2

Enter number of objects to add: 4


SUCCESS: Objects added successfully

Available objects (4):
1. Descriptor: 0
2. Descriptor: 1
3. Descriptor: 2
4. Descriptor: 3

Would you like to return to menu? (Y | N): y


Choose one option from the menu below
1. Show objects
2. Add objects
3. Remove objects
4. Exit
Enter your choice: 3


Available objects (4):
1. Descriptor: 0
2. Descriptor: 1
3. Descriptor: 2
4. Descriptor: 3

Enter object number to remove: 3
SUCCESS: Object succesfully removed

Available objects (3):
1. Descriptor: 0
2. Descriptor: 1
3. Descriptor: 3

Would you like to return to menu? (Y | N): n

Would you like to continue program execution? (Y | N): n

Thanks for using this program
```

Код бібліотеки – lib.h

```
#include <bits/stdc++.h>
#include "lib.h"
using namespace std;

#define ll long long
#define all(x) (x).begin(), (x).end()
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define endl "\n"
void dbg_out()
{
    cerr << endl;
}
template <typename Head, typename... Tail>
void dbg_out(Head H, Tail... T)
{
    cerr << ' ' << H;
    dbg_out(T...);
}
#define dbg(...) cerr << "(" << #__VA_ARGS__ << "):",
dbg_out(__VA_ARGS__)

void bad(const string &INPUT)
{
    stringstream ss;
    ss << "\033[1;31mERROR: " << INPUT << "\033[0m";
    cerr << ss.str() << endl;
}

void good(const string &INPUT)
{
    stringstream ss;
    ss << "\033[1;32mSUCCESS: " << INPUT << "\033[0m";
    cerr << ss.str() << endl;
}

ll getNum()
{
    ll number;
    while (!(cin >> number))
    {
        cin.clear();
```

```cpp
        cin.ignore(256, '\n');
        cout << endl;
        bad("Enter an integer");
        cout << endl;
    }
    return number;
}

ostream &BOLD(ostream &os)
{
    return os << "\e[1m";
}
ostream &UNBOLD(ostream &os)
{
    return os << "\e[0m";
}

ostream &RED(ostream &os)
{
    return os << "\033[1;31m";
}
ostream &UNRED(ostream &os)
{
    return os << "\033[0m";
}

ostream &GREEN(ostream &os)
{
    return os << "\033[1;32m";
}
ostream &UNGREEN(ostream &os)
{
    return os << "\033[0m";
}

ostream &GRAY(ostream &os)
{
    return os << "\033[1;30m";
}
ostream &UNGRAY(ostream &os)
{
    return os << "\033[0m";
}

ostream &YELLOW(ostream &os)
{
    return os << "\033[1;33m";
}
```

```cpp
ostream &UNYELLOW(ostream &os)
{
    return os << "\033[0m";
}

ll showMenu(const vector<string> &MENU_OPTIONS)
{
    cout << BOLD << "Choose one option from the menu below\n"
         << UNBOLD;
    for (ll i = 0; i < MENU_OPTIONS.size(); i++)
    {
        cout << i + 1 << ". " << MENU_OPTIONS[i] << endl;
    }
    cout << "Enter your choice: ";
    ll userDecision = getNum();
    cout << endl;
    return userDecision;
}

string validateName(string inputString)
{
    stringstream stringProcessor(inputString);
    string wordHolder;
    string resultHolder;
    while (stringProcessor >> wordHolder)
    {
        if (!isupper(wordHolder[0]))
        {
            wordHolder[0] = toupper(wordHolder[0]);
        }
        resultHolder += wordHolder + " ";
    }
    return resultHolder;
}

string getEmailAddress()
{
    string emailAddress;
    cout << "Please enter an email address: ";
    cin >> emailAddress;
    if (emailAddress.find("@") == string::npos)
    {
        cout << "\nERROR: Invalid email address\n\n";
        getEmailAddress();
    }
    else
        return emailAddress;
    return "";
```

```
}

string getFileName()
{
    string fileName = "";
    bool isExtensionFound = true;
    do
    {
        cout << "Enter file name: ";
        cin >> fileName;

        if (fileName.find(".") == string::npos)
        {
            isExtensionFound = false;
            cout << "\nERROR: File extension not found. Try
again...\n\n";
            continue;
        }
        else
            break;
    } while (isExtensionFound == false);
    return fileName;
}

string generateRandomString(int length)
{
    string chars = "abcdefghijklmnopqrstuvwxy";
    string randomString = "";
    for (int i = 0; i < length; i++)
    {
        int index = rand() % chars.size();
        randomString += chars[index];
    }
    return randomString;
}

string generateRandomPassword(int length)
{
    string chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxy1234567890!@#$%^&*()_+
=-[]{}`~';/.,";
    string randomPass = "";
    for (int i = 0; i < length; i++)
    {
        int index = rand() % chars.size();
        randomPass += chars[index];
    }
    return randomPass;
```

```cpp
}

template <typename T>
vector<T> getUniqueVector(vector<T> &inputVector)
{
    vector<T> uniqueElements;
    unordered_set<T> seenElements;
    for (T element : inputVector)
    {
        if (seenElements.find(element) == seenElements.end())
        {
            uniqueElements.push_back(element);
            seenElements.insert(element);
        }
    }
    return uniqueElements;
}

template <typename T>
void quickSort(vector<T> &arr, int left, int right)
{
    int i = left, j = right;
    int pivot = arr[(left + right) / 2];

    if (arr.size() <= 1)
        return;

    while (i <= j)
    {
        while (arr[i] > pivot)
            i++;
        while (arr[j] < pivot)
            j--;

        if (i <= j)
        {
            swap(arr[i], arr[j]);
            i++;
            j--;
        }
    };

    if (left < j)
        quickSort(arr, left, j);
    if (i < right)
        quickSort(arr, i, right);
}
```

```cpp
template <typename T>
void exchangeSort(vector<T> &arr)
{
    if (arr.size() <= 1)
        return;

    for (int i = 0; i < arr.size() - 1; i++)
        for (int j = i + 1; j < arr.size(); j++)
            if (arr[i] < arr[j])
                swap(arr[i], arr[j]);
}

template <typename T>
void bubbleSort(vector<T> &arr)
{
    if (arr.size() <= 1)
        return;

    for (int i = 0; i < arr.size(); i++)
        for (int j = 0; j < arr.size() - i - 1; j++)
            if (arr[j] < arr[j + 1])
                swap(arr[j], arr[j + 1]);
}

template <typename T>
void mergeSort(vector<T> &arr)
{
    if (arr.size() <= 1)
        return;

    vector<int> left, right;
    int middle = arr.size() / 2;

    for (int i = 0; i < middle; i++)
        left.push_back(arr[i]);
    for (int i = middle; i < arr.size(); i++)
        right.push_back(arr[i]);

    mergeSort(left);
    mergeSort(right);

    int i = 0, j = 0, k = 0;

    while (i < left.size() && j < right.size())
    {
        if (left[i] > right[j])
        {
            arr[k] = left[i];
            i++;
        }
        else
        {
            arr[k] = right[j];
            j++;
        }
        k++;
    }

    while (i < left.size())
    {
        arr[k] = left[i];
        i++;
        k++;
    }

    while (j < right.size())
    {
        arr[k] = right[j];
        j++;
        k++;
    }
}

template <typename T>
void outputArray(vector<T> arr)
{
    for (auto i : arr)
        cout << i << " ";
    cout << endl;
}

void outputArray(int *arr)
{
    int n = sizeof(arr) / sizeof(int);
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

template <typename T>
void outputArray(vector<vector<T>> &arr)
{
    int n = arr.size();
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
```

```cpp
            cout << arr[i][j] << " ";
        cout << "\n";
    }
}

string toLower(string str)
{
    transform(str.begin(), str.end(), str.begin(), ::tolower);
    return str;
}
```

Висновки

Таким чином, ми навчилися використовувати шаблони при створенні програм мовою програмування С++.