**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №8

з дисципліни «Об’єктно-орієнтоване програмування»

На тему «ОБРОБКА ВИНЯТКОВИХ СИТУАЦІЙ»

Варіант №20

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Ст. Викладач Т. В. Голуб

Ст. Викладач Л. Ю. Дейнега

2023

[Мета роботи 3](#_Toc133148955)

[Текст завдання №5 3](#_Toc133148956)

[Код програми – main.cpp 3](#_Toc133148957)

[Код програми – sup.h 4](#_Toc133148958)

[Код програми – exc.h 10](#_Toc133148959)

[Код бібліотеки – lib.h 11](#_Toc133148960)

[Приклад роботи 19](#_Toc133148961)

[Висновки 19](#_Toc133148962)

Мета роботи

Навчитись обробляти виняткові ситуації при створені програм мовою програмування С++.

Текст завдання №5

Для завдання з лабораторної роботи №5 виконати обробку виняткових ситуацій з використанням класу Exception.

Код програми – main.cpp

// include necessary libraries

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

#include "sup.h"

#include "exc.h"

using namespace std;

// Для завдання з лабораторної роботи #5 виконати обробку виняткових ситуацій з використанням класу Exception.

// func main start

int main()

{

    // declare local variables //

    srand(time(NULL));

    char doContinue;

    /////////////////////////////

    // project intro

    cout << "\n";

    do

    {

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

        // for storing class object pointers

        vector<DynamicString> container;

        // for manipulating program flow

        char doReturnToMenu;

        do

        {

            // output menu to user

            outputMenu(container);

            // ask user if they would like to return to menu

            cout << "\nWould you like to return to menu? (Y | N): ";

            cin >> doReturnToMenu;

            // if so, continue loop execution

            if (doReturnToMenu == 'Y' || doReturnToMenu == 'y')

            {

                cout << endl

                     << endl;

                continue;

            }

            // if not, break out of loop

            else

                break;

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

        // execute while user chooses to return to menu

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

        // 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;

}

Код програми – sup.h

#include "sup.h"

#include "exc.h"

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

const string ROOT\_DIR = "D:/repos/university/year1-term2/OOP/lb8/";

class DynamicString

{

private:

    char \*strValue;

    size\_t strSize;

public:

    DynamicString() : strValue(nullptr), strSize(0) {}

    DynamicString(const char \*INPUT) : strValue(nullptr), strSize(0)

    {

        if (INPUT)

        {

            strSize = strlen(INPUT) + 1;

            strValue = new char[strSize];

            strcpy\_s(strValue, strSize, INPUT);

        }

    }

    DynamicString(const DynamicString &other)

    {

        size\_t len = strlen(other.strValue) + 1;

        strValue = new char[len];

        strcpy\_s(strValue, len, other.strValue);

    }

    DynamicString &operator=(const char \*INPUT)

    {

        delete[] strValue;

        size\_t inputSize = strlen(INPUT) + 1;

        strValue = new char[inputSize];

        strcpy\_s(strValue, inputSize, INPUT);

        return \*this;

    }

    DynamicString &operator=(const DynamicString &INPUT)

    {

        delete[] strValue;

        strSize = INPUT.strSize;

        strValue = new char[strSize + 1];

        strcpy(strValue, INPUT.strValue);

        return \*this;

    }

    friend ostream &operator<<(ostream &outputStream, const DynamicString &OUTPUT)

    {

        outputStream << OUTPUT.strValue;

        return outputStream;

    }

    friend istream &operator>>(istream &inputStream, DynamicString &inputHolder)

    {

        char buffer[65536];

        inputStream.getline(buffer, 65536);

        inputHolder = buffer;

        return inputStream;

    }

    friend ofstream &operator<<(ofstream &outputStream, const DynamicString &OUTPUT)

    {

        outputStream << OUTPUT.strValue;

        return outputStream;

    }

    friend ifstream &operator>>(ifstream &inputStream, DynamicString &inputHolder)

    {

        char buffer[65536];

        inputStream.getline(buffer, 65536);

        inputHolder = buffer;

        return inputStream;

    }

    ~DynamicString()

    {

        delete[] strValue;

    }

};

void showStrings(vector<DynamicString> &container)

{

    ll stringsNum = container.size();

    try

    {

        if (stringsNum == 0)

        {

            throw IOException();

        }

    }

    catch (IOException &e)

    {

        bad(e.what());

        exit(1);

    }

    cout << "Available strings (" << stringsNum << "):\n";

    for (ll i = 0; i < stringsNum; i++)

    {

        cout << i + 1 << ". " << container[i] << endl;

    }

}

void showStrings(vector<DynamicString> &container, const string &FILE)

{

    ll stringsNum = container.size();

    try

    {

        if (stringsNum == 0)

        {

            throw IOException();

        }

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    ofstream file(FILE);

    if (!file.is\_open())

    {

        throw IOException();

    }

    file << "==============================\n\n";

    file << "Available strings (" << stringsNum << "):\n";

    for (ll i = 0; i < stringsNum; i++)

    {

        file << i + 1 << ". " << container[i] << endl;

    }

    file << "\n==============================\n\n";

    file.close();

    try

    {

        if (file.good())

            good("Strings succesfully saved");

        else

            throw IOException();

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

}

void addStrings(vector<DynamicString> &container)

{

    ll initSize = container.size();

    cout << "Enter number of strings to add: ";

    ll numToAdd = getNum();

    cout << endl;

    cin.ignore();

    try

    {

        if (numToAdd < 1)

        {

            throw IOException();

        }

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    for (ll i = 0; i < numToAdd; i++)

    {

        DynamicString value;

        cout << i + 1 << ". Enter value: ";

        cin >> value;

        container.eb(value);

    }

    cout << endl;

    try

    {

        if (container.size() == initSize + numToAdd)

            good("Strings succesfully added");

        else

            throw IOException();

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    cout << endl;

    showStrings(container);

}

void addStrings(vector<DynamicString> &container, const string &FILE)

{

    ll initSize = container.size();

    ifstream file(FILE);

    string line;

    vector<string> lines;

    try

    {

        if (!file.is\_open())

        {

            throw IOException();

        }

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    while (getline(file, line))

    {

        lines.push\_back(line);

    }

    file.close();

    for (ll i = 0; i < lines.size(); i++)

    {

        DynamicString stringHolder = lines[i].c\_str();

        container.eb(stringHolder);

    }

    try

    {

        if (container.size() == initSize + lines.size())

            good("Strings succesfully added");

        else

            throw IOException();

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

}

void removeString(vector<DynamicString> &container)

{

    ll initSize;

    try

    {

        initSize = container.size();

        if (initSize == 0)

        {

            throw IOException();

        }

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    initSize = container.size();

    cout << endl;

    showStrings(container);

    cout << endl;

    cout << "Enter number of string to remove: ";

    ll numToRemove = getNum();

    numToRemove--;

    try

    {

        if (numToRemove < 0 || numToRemove >= initSize)

        {

            throw IOException();

        }

    }

    catch (Exception &e)

    {

        bad(e.what());

        exit(1);

    }

    container.erase(container.begin() + numToRemove);

    cout << endl;

    try

    {

        if (container.size() == initSize - 1)

            good("String succesfully removed");

        else

            throw IOException();

    }

    catch (const Exception &e)

    {

        bad(e.what());

        exit(1);

    }

}

void outputMenu(vector<DynamicString> &container)

{

    vector<string> menuItems = {

        "Show strings",

        "Add strings",

        "Remove strings",

        "Exit"};

    ll userDecision = showMenu(menuItems);

    if (userDecision == 1)

    {

        menuItems = {

            "Output to console",

            "Output to file",

            "Exit"};

        userDecision = showMenu(menuItems);

        if (userDecision == 1)

        {

            showStrings(container);

        }

        else if (userDecision == 2)

        {

            string fileName = ROOT\_DIR;

            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\_DIR;

            fileName += getFileName();

            addStrings(container, fileName);

        }

    }

    else if (userDecision == 3)

    {

        removeString(container);

    }

}

Код програми – exc.h

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

#include "exc.h"

/\*

Створити  базовий  клас  Exception,  та  відповідні  класи-спадкоємці, що дозволяють обробляти наступні виняткові ситуації:

    а)  помилки  при  роботі  з  потоками  введення/виведення, зокрема при роботі з  файлами;

    б)  помилки арифметичних операцій (ділення на 0);

    в)  помилки  виділення  динамічної  пам’яті  при перевантаженні операторів new  та delete.

\*/

class Exception

{

public:

    virtual void what() const throw()

    {

        bad("An exception has occurred");

    }

};

class IOException : public Exception

{

public:

    void what() const throw() override

    {

        bad("I/O stream error");

    }

};

class ArithmeticException : public Exception

{

public:

    void what() const throw() override

    {

        bad("Arithmetic error: division by zero");

    }

};

class MemoryException : public Exception

{

public:

    void what() const throw() override

    {

        bad("Memory allocation error");

    }

};

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();

        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";

}

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;

}

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);

}

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++)

            cout << arr[i][j] << " ";

        cout << "\n";

    }

}

string toLower(string str)

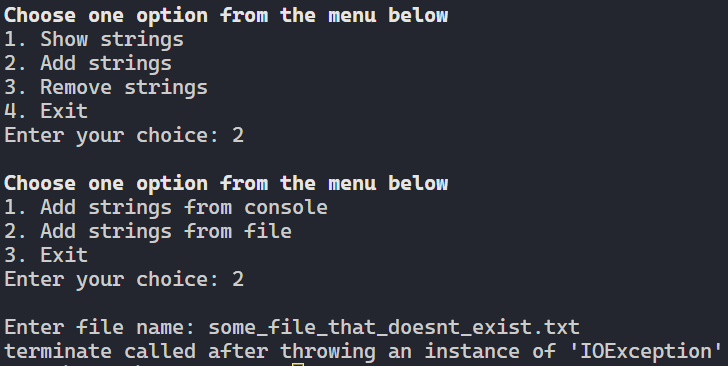
{

    transform(str.begin(), str.end(), str.begin(), ::tolower);

    return str;

}

Приклад роботи



Висновки

Таким чином, ми навчилися обробляти виняткові ситуації при створенні програм мовою програмування С++.