Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 11

з навчальної дисципліни “Базові методології та технології

програмування ”

**Реалізація програмних засобів оброблення динамічних структур даних та бінарних файлів**

ВИКОНАВ

студент академічної групи КІ-20

Микитенко Д. Ю

Середа Є. В.

Гарбар В. В.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

Л.І. Поліщук

Кропивницький – 2021

Мета роботи: полягає у набутті грунтовних вмінь і практичних навичок командної(колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів С++ для керування динамічною пам’яттю та бінарними файловими потоками.

Завдання до лабораторної роботи:

1.  У складі команди IT-проєкта розробити програмні модулі оброблення динамічної структури даних.

2. Реалізувати програмний засіб на основі розроблених командою IT-проекта модулів.

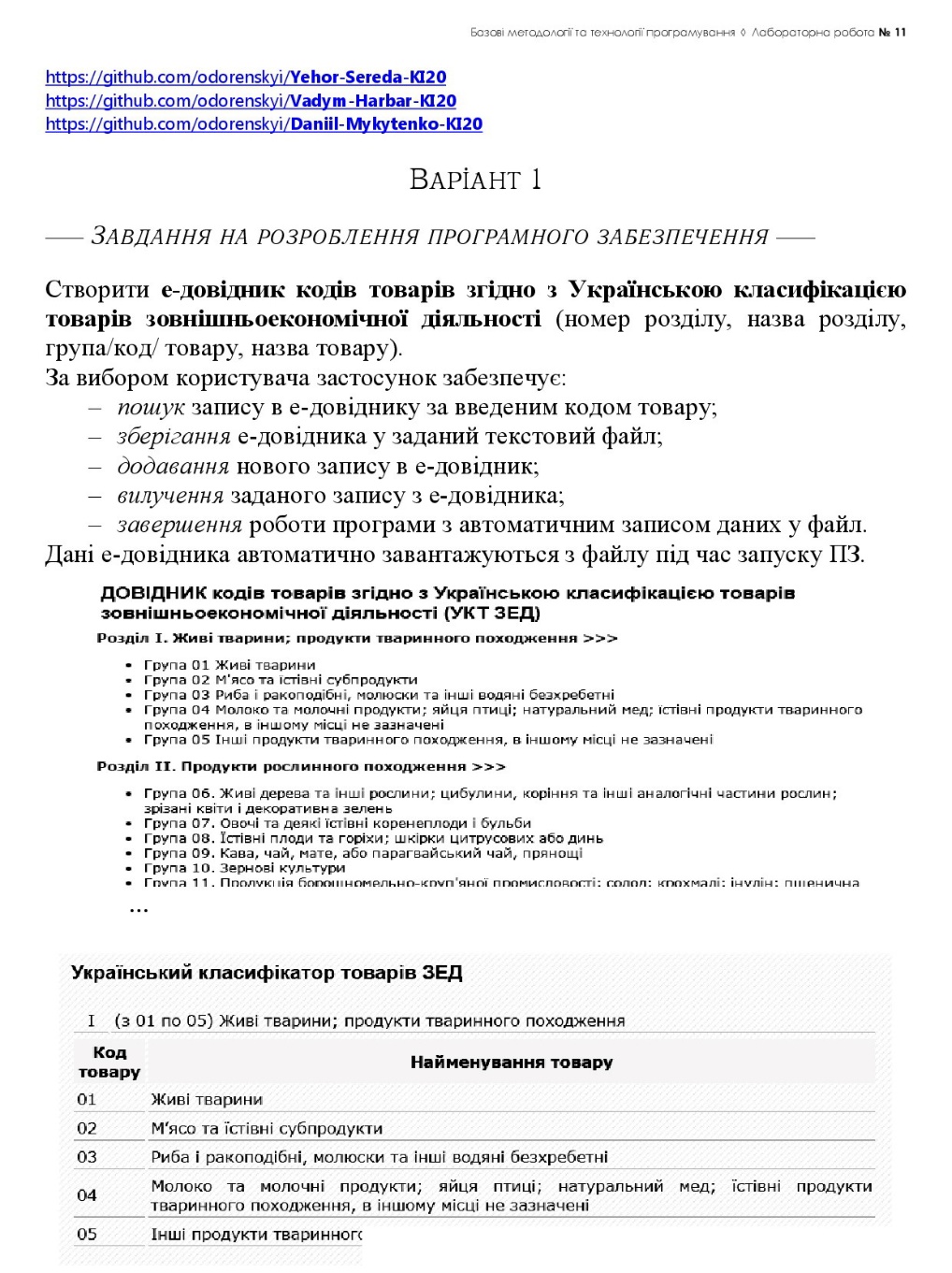


Рисунок 1 – Завдання за варіантом 1

Задача 11.1

**Формалізація задачі:**

**Вихідні змінні:** оброблений текст вхідного тексту методами.

**Вхідні змінні:** Українська класифікація товарів зовнішньоекономічної діяльності(номер розділу/назва розділу/група/код товару/назва товару) (УКТ ЗЕД).

**Вимоги до ПЗ:**

1. Інтерефейс українською мовою.
2. Реалізація усіх методів.

**Алгоритм виконання:**

1. Проектування архітектури та розподіл праці між командою.
2. Написання функцій членами команди.
3. Розподіл формальних вимог(інтерфейс, тестування, звіт) між командою.
4. Об’єднання праць команди.

Висновки

Під час виконання лабораторної роботи №11 було здійснено аналіз і постановку задачі 11.1, одержані дані було додано до звіту. Згодом було розроблено набір контрольних прикладів до задачі задля виконання тестування, усі отримані тест-кейси були додані до звіту як додаток. У QT\_Creator створено проєкт майбутнього проєкту. Були розподілені функції між членами команди. Микитенко Даниїл – фундамент проєкту, метод пошуку, метод зберігання у файл, написання звіту; Гарбар Вадим – метод пошуку, метод вилучення, написання інтерфейсу та тест-сьютів; Середа Єгор – метод пошуку, метод додавання, метод вилучення. Після цього функції були об’єднанні та протестовані у main. Результати співпадали з дійсністю.

Усі тест-кейси мали результат виконання Passed, тому не було потреби виконувати від лагодження програми. Усі результати тестування були задокументовані та додані до звіту як додаток. Вихідний код проєкту MykytenkoSeredaHarbar\_task включено до звіту як додаток.

Після вдалої компіляції проекту було здійснено системне тестування ПЗ за допомогою раніше розроблених тест-сьютів. Не було виявлено негативних статусів під час тестування, тому виконувати відлагодження програми не знадобилося.

Отриманий тестовий артефакт було збережено й додано до звіту як додаток. Консольний додаток було скопійовано у теку Software.

Я вважаю, що дана лабораторна робота є дуже корисною, перш за все ми продовжуємо користуватися GitHub, навички роботи з яким є досить корисними у подальшій роботі з командою програмістів і найголовніше – ми почали працювати у команді й почали навчатися виконувати проєкт у команді. Друге, це те що ми почали використовувати, а також оновлювати й додавати нові функції до статичної бібліотеки, яка є досить зручним, якщо часто використувати їх у своїх лабораторних роботах і не тільки. По-третє, це ще й модульне тестування за допомогою написання тестових драйверів власноруч, те що тестові драйвери не є шаблоними, це правильно, тому що при їх написанні можна одразу знайти недоліки програми й внести зміни. Й в-четверте, ми навчилися працювати з файлами, потоками введення в них та виведення з них, на мою думку, це одна з найкорисніших тем.

Найголовнішими проблемами під час виконання лабораторної роботи №11 були наступні проблеми:

1. Проєктування архітектури. Це було доволі складно, адже у кожного члену команди своє бачення виконання проєкту. Згодом ми доволі швидко дійшли до спільного рішення.
2. Проблеми під час об’єднання функцій, адже у кожного свій підхід. Але це було надзвичайно корисно, адже ми навчились адаптуватись під інший стиль коду та навчились оптимізовувати інший код.

Також під час роботи з проєктом, часто виникали якісь проблеми, через що приходилось виправляти код. Це було надзвичайно зручно, адже увесь код функцій був прописаний у статичній бібліотеці Modules\_Mykytenko, через що знайти потрібну частину коду й працювати з нею окремо було набагато краще, ніж пошук усього цього в main.

Тепер зрозуміло, що кожну програму можна розбити на команду та розроблювати кожну функцію окремо, я вважаю це правильно, тому що при розробці великих проектів, розробник повинен поділяти свою програму на методи, задля зручного перегляду та відлагодження, а також, якщо цей проект виконується командою, модульне програмування є необхідним.

Також було непогано попрацювати з масивами, векторами й структурами, закріпити в пам’яті роботу з ними.

Для мінімалізації коду та максимальної його оптимізації в модулі також було розроблено декілька функцій для вводу чисел, а сам при викликі головних функцій, як аргумент передавались не змінні, а самі функції вводу.

Додаток 1. Лістинг модуля QTModulesMykytenko.h

#include <vector>

#include <iostream>

#include <fstream>

#include <wchar.h>

#include <fcntl.h>

#include <clocale>

#include <windows.h>

#include <codecvt>

#include <cctype>

#include <ctime>

#include <math.h>

using namespace std;

wstring stages[5] = {L"Розділ 1; Живі тварини; продукти тваринного походження",

L"Розділ 2; Продукти рослинного походження",

L"Розділ 3; Жири та олії тваринного або рослинного походження; продукти їх розщеплення; готові харчові жири; воски тваринного або рослинного походження",

L"Розділ 4; Готові харчові продукти; алкогольні та безалкогольні напої і оцет; тютюн та його замінники",

L"Розділ 5; Мінеральні продукти"};

wstring section01[5] = {L"\tГрупа 1 - Живі тварини", L"\tГрупа 2 - М\'ясо та їстівні субпродукти", L"\tГрупа 3 - Риба та їстівні субпродукти",

L"\tГрупа 4 - Молоко та молочні продукти; їстівні продукти тваринного походження, в іншому місці не зазначені",

L"\tГрупа 5 - Інші продукти тваринного походження, в іншому місці не зазначені"};

wstring section02[6] = {L"\tГрупа 6 - Живі дерева та інші рослини; цибулини, коріння та інші аналогічні частини рослин",

L"\tГрупа 7 - Овочі та деякі їстівні коренеплоди і бульби",

L"\tГрупа 8 - Їстівні плоди та горіхи; шкірки цитрусових або динь",

L"\tГрупа 9 - Кава, чай, мате, або парагвайський чай, прянощі",

L"\tГрупа 10 - Зернові культури",

L"\tГрупа 11 - Продукція борошномельно-круп\'яної промисловості; солод, крохмалі; інулін; пшенична клейковина"};

wstring section03[1] = {L"\tГрупа 12 - Жири та олії тваринного або рослинного походження; продукти їх розщеплення; готові харчові жири; воски тваринного або рослинного походження"};

wstring section04[9] = {L"\tГрупа 13 - Готові харчові продукти з м'яса, риби або ракоподібних, молюсків або інших водяних безхребетних",

L"\tГрупа 14 - Цукор і кондитерські вироби з цукру; Какао та продукти з нього",

L"\tГрупа 15 - Готові продукти із зерна зернових культур, борошна, крохмалю або молока; борошняні кондитерські вироби",

L"\tГрупа 16 - Продукти переробки овочів, плодів або інших частин рослин",

L"\tГрупа 17 - Різні харчові продукти; Алкогольні і безалкогольні напої та оцет",

L"\tГрупа 18 - Залишки і відходи харчової промисловості; готові корми для тварин",

L"\tГрупа 19 - Тютюн і промислові замінники тютюну"};

wstring section05[3] = {L"\tГрупа 20 - Сіль; сірка; землі та каміння; штукатурні матеріали, вапно та цемент",

L"\tГрупа 21 - Руди, шлаки та зола",

L"\tГрупа 22 - Палива мінеральні; нафта та продукти її перегонки; бітумінозні речовини; воски мінеральні"};

struct **product** //глобальная переменная структуры с товаром

{

int code;

wstring name;

};

struct **section** //глобальная переменная структуры с разделами

{

wstring groupName;

int groupPos;

vector<struct product> groups;

};

vector<struct section> sections; //глобальная переменная вектора разделов

vector<wstring> section1;

vector<wstring> section2;

vector<int> groupsAmount;

wstring **nameGetter**(wstring place) //возвращаем имя группы или имя товара(в зависимости от аргумента)

{

wstring name;

wcout << L"Введіть назву " << place << L": ";

wcin >> name;

return name;

}

int **posGetter**(wstring place) //возвращаем код группы или позицию товара(в зависимости от аргумента)

{

int position;

wcout << L"Введіть номер " << place << L": ";

wcin >> position;

return position;

}

product **productsGetter**(int iter) //возвращаем структуру

{

product example;

example.code = iter;

example.name = nameGetter(L"товару");

return example;

}

product **defaultGroup**(wstring name, int iter)

{

product example;

example.code = iter;

example.name = name;

return example;

}

void **clean\_input**()

{

fflush(stdin);

wcin.clear();

}

wstring **find\_group\_by\_code**(int code) //костыль тот же, но это работает правильно, теперь точно

{

/\*Входной параметр функции - код группы.

Поиск осуществляется по функции поиска подстроки.\*/

wifstream output;

output.open("result.txt");

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*)); //локализация файлового потока

wstring line;

if(code == 1)

{

for(int i = 0; i < 2; i++)

getline(output, *line*);

return line;

}

while(getline(output, *line*))

{

if(wcsstr(line.c\_str(), to\_wstring(code).c\_str())) break; //Если номер группы присутствует в строке, тогда присвоить его результирующему стрингу

}

output.close();

return line;

}

int **defaultFiller**(int iterator)

{

wofstream output; //создание объекта для записи в файл

output.open("result.txt");

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*)); //локализация файлового потока

for(int i = 0; i < 5; i++)

{

section1.push\_back(section01[i]);

}

for(int i = 0; i < 6; i++)

{

section2.push\_back(section02[i]);

}

section A;

A.groupName = stages[0];

A.groupPos = groupsAmount.size() + 1;

for(unsigned long long i = 0; i < section1.size(); i++) //инициализация раздела 1 дефолтными значениями

{

iterator++;

A.groups.push\_back(defaultGroup(section1[i], iterator));

}

sections.push\_back(A); //возвращаем структуру раздела 1 в вектор

groupsAmount.push\_back(groupsAmount.size()+1);

output << A.groupName << endl;

for(unsigned long long i = 0; i < section1.size(); i++)

output << section01[i] << endl;

section B; //структура раздела 2

B.groupName = stages[1];

B.groupPos = groupsAmount.size()+1;

for(unsigned long long i = 0; i < section2.size(); i++) //инициализация раздела 1 дефолтными значениями

{

iterator++; //итератор мы тоже не трогаем, он работает как надо

B.groups.push\_back(defaultGroup(section2[i], iterator));

}

sections.push\_back(B); //возвращаем структуру раздела 2 в вектор

groupsAmount.push\_back(groupsAmount.size()+1);

output << B.groupName << endl;

for(unsigned long long i = 0; i < section2.size(); i++)

output << section02[i] << endl;

return iterator;

}

int **indexGetter**(int a) // функция для определения индекса строки(для итерационного поиска, не трогай, всё работает)

{

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

input.open("result.txt");

wstring cop;

int index = 0;

while(getline(input,*cop*))

{

index++;

if(cop == find\_group\_by\_code(a))

break;

}

return index;

}

void **delete\_string**(int a) //функция ясночего(не трогай, всё работает)

{

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

input.open("result.txt");

wofstream memory;

memory.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

memory.open("temp.txt");

wstring cop, group\_str = L"\tГрупа ";

for(int i = 0; getline(input,*cop*); i++)

{

for(int j=0;j<5;j++)

if(cop==stages[j]){

memory<<cop<<endl;

continue;

}

if(wcsstr(cop.c\_str(),group\_str.c\_str()))

if(i!=indexGetter(a)-1)

memory<<cop<<endl;

}

input.close();

memory.close();

remove("result.txt");

rename("temp.txt","result.txt");

}

wstring **show\_file**() //функция вывода содержимого файла

{

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

input.open("result.txt");

wstring line;

wstring copys;

if(!input.is\_open()){

wcout<<L"Файл не існує, або не вдалося знайти";

}

while (getline(input, *line*))

{

copys+=line;

copys.push\_back('\n');

}

input.close();

return copys;

}

wstring **add\_group\_to\_section**(wstring data, wstring \*groups, int size\_of\_arr) //функция добавления группы к новому раздел

{

int number\_of\_group = 0, stored\_values[size\_of\_arr], iter = 0, flag = 0;

wchar\_t key;

do

{

for(int i = 0; i < size\_of\_arr; i++)

{

if(i != stored\_values[i]) wcout << groups[i] << endl;

}

wcout << L"Введіть номер групи, що ви бажаєте додати від 1 до " + to\_wstring(size\_of\_arr) + L':';

wcin >> number\_of\_group; //нужна валидация

for(int i = 0; i < size\_of\_arr; i++)

{

if(number\_of\_group == stored\_values[i])

{

wcout << L"Помилка додавання групи! Дана група вже присутня у файлі." << endl;

flag++;

break;

}

}

if(iter < size\_of\_arr && flag == 0)

{

if(!(wcsstr(data.c\_str(), groups[number\_of\_group - 1].c\_str())))

{

stored\_values[iter] += number\_of\_group;

data += L'\n' + groups[number\_of\_group - 1];

wcout << L"Група була успішно добавлена!" << endl;

}

else

{

wcout << L"Помилка додавання групи! Дана група вже присутня у файлі." << endl;

}

}

wcout << L"Чи бажаєте ви додати ще одну группу?(y = так, n = ні):";

wcin >> key; //нужна валидация

iter++;

}

while(key != L'n');

return data;

}

int **add\_section**() //функция добавления раздела

{

wstring line, data;

int number;

wcout << L"Введіть номер розділу, що ви бажаєте додати до файлу:";

while(!(wcin >> number) || (number < 1 || number > 5))

{

wcout << L"Помилка додавання розділу. Будь-ласка, введіть число від 1 до 5" << endl;

clean\_input();

}

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*)); //локализация файлового потока

input.open("result.txt");

while(getline(input, *line*))

{

data += line;

data.push\_back(L'\n');

}

input.close();

if(!(wcsstr(data.c\_str(), stages[number-1].c\_str())))

{

data += stages[number-1];

}

else

{

wcout << L"Помилка додавання розділу! Даний розділ вже присутній у файлі.";

return 0;

}

switch(number)

{

case 1: data = L'\n' + add\_group\_to\_section(data, *section01*, 5) + L'\n'; break;

case 2: data = L'\n' + add\_group\_to\_section(data, *section02*, 6) + L'\n'; break;

case 3: data = L'\n' + add\_group\_to\_section(data, *section03*, 1) + L'\n'; break;

case 4: data = L'\n' + add\_group\_to\_section(data, *section04*, 9) + L'\n'; break;

case 5: data = L'\n' + add\_group\_to\_section(data, *section05*, 3) + L'\n'; break;

default: break;

}

wofstream output;

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

output.open("result.txt");

output << data;

output.close();

return 1;

}

void **add\_group**() // функция добавления группы в файл

{

int number, number\_of\_group, size\_arr, occurrences = 0;

wstring \*arr, line, data;

wstring::size\_type pos = 0;

wcout << L"Введіть номер розділу, з якого ви бажаєте взяти группу:";

while(!(wcin >> number) || (number < 1 || number > 5))

{

wcout << L"Помилка додавання розділу. Будь-ласка, введіть число від 1 до 5" << endl;

clean\_input(); //функция очистки буфера от хлама

}

switch(number)

{

case 1: arr = section01; size\_arr = 5; break;

case 2: arr = section02; size\_arr = 6; break;

case 3: arr = section03; size\_arr = 1; break;

case 4: arr = section04; size\_arr = 9; break;

case 5: arr = section05; size\_arr = 3; break;

default: break;

}

wcout << L"Введіть номер группи від 1 до " + to\_wstring(size\_arr) + L':';

while(!(wcin >> number\_of\_group) || (number\_of\_group < 1 || number\_of\_group > size\_arr))

{

wcout << L"Помилка! Невірні вхідні дані. Введіть номер групи, що ви бажаєте додати від 1 до " + to\_wstring(size\_arr) + L':';

clean\_input();

}

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

input.open("result.txt");

while(getline(input, *line*))

{

data += line;

if(line == arr[number\_of\_group-2])

{

data.push\_back(L'\n');

data += arr[number\_of\_group-1];

}

data.push\_back(L'\n');

}

input.close();

while ((pos = data.find(arr[number\_of\_group-1], pos )) != string::npos)

{

++ occurrences;

pos += arr[number\_of\_group-1].length();

wcout << occurrences;

}

if(occurrences == 1)

{

wofstream output;

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

output.open("result.txt");

output << data;

output.close();

}

else

wcout << L"Помилка, дана група вже присутня в файлі!" << endl;

}

void **delete\_section**()

{

wstring data, line, delete\_str, \*arr;

int size\_arr, number;

wifstream input;

input.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

input.open("result.txt");

wcout << L"Введіть номер розділу, який ви бажаєте видалити:";

while(!(wcin >> number) || (number < 1 || number > 5))

{

wcout << L"Помилка додавання розділу. Будь-ласка, введіть число від 1 до 5:";

clean\_input(); //функция очистки буфера от хлама

}

switch(number)

{

case 1: arr = section01; size\_arr = 5; break;

case 2: arr = section02; size\_arr = 6; break;

case 3: arr = section03; size\_arr = 1; break;

case 4: arr = section04; size\_arr = 9; break;

case 5: arr = section05; size\_arr = 3; break;

default: break;

}

while(getline(input, *line*))

{

data += line;

data.push\_back(L'\n');

}

input.close();

data.erase(data.find(stages[number - 1] + L'\n'), delete\_str.size());

for(int i = 0; i < size\_arr; i++)

{

delete\_str = arr[i] + L'\n';

data.erase(data.find(arr[i]), delete\_str.size());

}

wofstream output;

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*));

output.open("result.txt");

output << data;

output.close();

}

void **fullfillDefault**()

{

wofstream output; //создание объекта для записи в файл

output.open("result.txt");

output.imbue(locale(locale(), new codecvt\_utf8\_utf16*<*wchar\_t*>*)); //локализация файлового потока

output << stages[0] << endl;

output << section01[0] << endl << section01[1] << endl << section01[2] << endl << section01[3] << endl << section01[4] << endl;

output << stages[1] << endl;

output << section02[0] << endl << section02[1] << endl << section02[2] << endl << section02[3] << endl << section02[4] << endl << section02[5] << endl;

output << stages[2] << endl;

output << section03[0] << endl;

output << stages[3] << endl;

output << section04[0] << endl << section04[1] << endl << section04[2] << endl << section04[3] << endl << section04[4] << endl << section04[5] << endl << section04[6] << endl << section04[7] << endl << section04[8];

output << stages[4] << endl;

output << section05[0] << endl << section05[1] << endl << section05[2] << endl;

output.close();

}

int **main**()

{

\_wsetlocale(LC\_ALL, L"uk\_UA.UTF-8");

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

\_setmode(\_fileno(stdin), \_O\_U16TEXT);

wchar\_t checkbox;

wcout<<L"Електроний довідник версія 1.0\nРозробники:Гарбар Вадим, Микитенко Даниїл,Єгор Середа.\n";

while(true){

wcout<<L"---------Виберіть дію----------\n";

wcout<<L"Введіть a для відображення довідника\n";

wcout<<L"Введіть b для видалення вмісту розділу\n";

wcout<<L"Для додавання групи до розділу введіть с\n";

wcout<<L"Введіть e для повного заповнення\n";

wcout<<L"Введіть q для повернення до початкового стану\n";

wcout<<L"Введіть d для додавання розділу із групами\n";

wcout<<L"Введіть v для пошуку групи\n";

wcout<<L"Введіть r для видалення групи\n";

wcout<<L"Введіть x для виходу із программи\n";

wcout<<L"==========>";

wcin>>checkbox;

switch (checkbox) {

case 'a':

{

wcout<<show\_file();

break;

}

case 'b':

{

delete\_section();

break;

}

case 'c':

{

add\_group();

break;

}

case 'q':

{

int iterator = defaultFiller(0);

break;

}

case 'd':

{

add\_section();

break;

}

case 'e':

{

fullfillDefault();

break;

}

case 'r':

{

int a=0;

wcout<<L"====>";

wcin>>a;

delete\_string(a);

break;

}

case 'v':

{

int a=0;

wcout<<L"====>";

wcin>>a;

wcout<<find\_group\_by\_code(a);

break;

}

case 'x':

{

return 0;

break;

}

}

}

system("pause");

return 0;

}

Додаток 2. Тестовий набір до задачі 11.1

|  |  |
| --- | --- |
| Назва тестового набору | TS11\_LAB\_11\_BMTP |
| Назва проекта | Electro\_dovidnik |
| Рівень тестування | Програмне тестування |
| Автор тест-сьюта | Гарбар В.В |

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Вхідні дані | Очікувані дані | Результат |
| TestCase\_1 | Ввести а | Відображення довідника у консолі | passed |
| TestCase\_2 | Введіть b | У файлі result буде видалено вміст розілу  Розділ 1; Живі тварини; продукти тваринного походження  Група 1 - Живі тварини  Група 2 - М'ясо та їстівні субпродукти  Група 3 - Риба та їстівні субпродукти  Група 4 - Молоко та молочні продукти; їстівні продукти тваринного походження, в іншому місці не зазначені  Група 5 - Інші продукти тваринного походження, в іншому місці не зазначені  Розділ 2; Продукти рослинного походження | passed |
| TestCase\_3 | Введіть c | Повне заповнення файлу | passed |
| TestCase\_4 | Введіть q для повернення до початкового стану | Розділ 1; Живі тварини; продукти тваринного походження  Група 1 - Живі тварини  Група 2 - М'ясо та їстівні субпродукти  Група 3 - Риба та їстівні субпродукти  Група 4 - Молоко та молочні продукти; їстівні продукти тваринного походження, в іншому місці не зазначені  Група 5 - Інші продукти тваринного походження, в іншому місці не зазначені  Розділ 2; Продукти рослинного походження  Група 6 - Живі дерева та інші рослини; цибулини, коріння та інші аналогічні частини рослин  Група 7 - Овочі та деякі їстівні коренеплоди і бульби  Група 8 - Їстівні плоди та горіхи; шкірки цитрусових або динь  Група 9 - Кава, чай, мате, або парагвайський чай, прянощі  Група 10 - Зернові культури  Група 11 - Продукція борошномельно-круп'яної промисловості; солод, крохмалі; інулін; пшенична клейковина | passed |
| TestCase\_5 | Введіть d | Додавання нового розділу із групами | passed |
| TestCase\_6 | Введіть v |  | passed |
| TestCase\_7 | Введіть r |  | passed |