**Лабораторна робота №13**

Тема: ***.* Програмування задач з використанням лінійних списків : стека, черги.**

**Мета:** **:**1.Ознайомлення з динамічною структурою даних -. стек, черга.

2.Отримання навиків роботи з змінними вказівного типу.

3.Ознайомлення з можливістю виконання операції, включення елементів в стек, чергу та їх виключення зі вказаного списку.

**ПОСТАНОВКА ЗАДАЧ ДО ЛР13.**

1. Виконати реалізацію програми згідно наданої викладачем таблиці варіантів: Задача 1.
2. Виконати реалізацію програми згідно наданої викладачем таблиці варіантів: Задача 2.

**3адача1: *Задачі підвищеної трудності на „відмінно” :10-12 балів.***

1. Реалізувати операцію додавання двох великих чисел зображених у формі стеків. Значенням елемента стеку буде значення певної цифри числа. Під час обчислення суми її розряди також слід записувати у стек, а потім – виводити значення елементів цього стеку.

**Код модуля ( модулів).**

#ifndef HEADER\_H

#define HEADER\_H

using namespace std;

struct stack{

int data;

stack\* next;

};

stack\* createnewstack(const string& a);

void print(stack\* head);

stack\* add(stack\* num1, stack\* num2);

int pop(stack\*& head);

void push(stack\*& head, int a);

#endif

#include <iostream>

#include "header.h"

#include<string>

#include <Windows.h>

using namespace std;

void push(stack\*& head, int a) {

stack\* newStack = new stack;

newStack->data = a;

newStack->next = head;

head = newStack;

}

int pop(stack\*& head){

if (head == NULL) {

cout << "Стек порожній" << endl;

return -1;

}

int a = head->data;

stack\* temp = head;

head = head->next;

delete temp;

return a;

}

stack\* add(stack\* num1, stack\* num2) {

stack\* result = NULL;

int dil = 0;

while (num1 != NULL || num2 != NULL || dil != 0) {

int a1 = (num1 != NULL) ? pop(num1) : 0;

int a2 = (num2 != NULL) ? pop(num2) : 0;

int suma = a1 + a2 + dil;

dil = suma / 10;

suma %= 10;

push(result, suma);

}

return result;

}

void print(stack\* head) {

while (head != NULL) {

cout << head->data;

head = head->next;

}

cout << endl;

}

stack\* createnewstack(const string&a) {

stack\* stk = NULL;

for(char d : a) {

int b = d - '0';

push(stk, b);

}

return stk;

}**Код основного модуля ( диспетчера – функції main())**

#include <iostream>

#include <Windows.h>

#include <string>

#include "header.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string num1, num2;

cout << "\n[1]>>> "; cin >> num1;

cout << "\n[2]>>> "; cin >> num2;

stack\* numb1 = createnewstack(num1);

stack\* numb2 = createnewstack(num2);

stack\* sum = add(numb1, numb2);

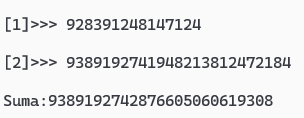
cout << "\nSuma:";

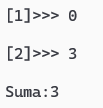
print(sum);

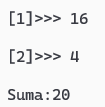
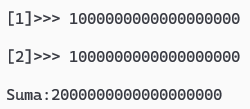
**Файли проекта:**

****

}**Тестовий набір вхідних даних. Результати роботи усіх функцій проєкту.**



****

**** ****

**Варіанти задачі 2**

***Задачі з використанням різних видів списків (на «відмінно»-10-12 балів)***

**Варіант №8**

Реалізувати список для зберігання і операцій з даними виду:

|  |  |  |  |
| --- | --- | --- | --- |
| Прізвище | Країна | Рік вступу | Вартість навчання |

У список входять усі записи. В чергу - лише ті записи, де поле "Країна" не дорівнює "Україна".У стек - лише ті записи, де поле Рік вступу більше заданого Р002E

Забезпечити виконання операцій:

* додавання нового елементу в невпорядкований список;
* впорядкування списку за полем "Прізвище";
* додавання елемента в впорядкований список;
* корегування поля "Країна" обраного елементу;
* роздрукування підсписків;
* роздрукування переліку країн.

**Код модуля ( модулів).**

#ifndef MODUL\_H

#define MODUL\_H

struct student {

char name[30];

char country[30];

int year;

double price;

};

struct mylist

{

student data;

mylist\* next;

};

struct stack

{

student data;

stack\* next;

};

struct cherga

{

student data;

cherga\* next;

};

void printStack(stack\* stackHead);

void printQueue(cherga\* queueHead);

void printList(mylist\* head);

mylist\* createList(cherga\*& chergaHead, stack\*& stackHead, int maxYear);

void addToUnorderedList(mylist\*& head, student newData, cherga\*& chergaHead, stack\*& stackHead, int maxYear);

void sortList(mylist\*& head);

void addToOrderedList(mylist\*& head, student newData);

//bool editCountry(mylist\* head, const char\* nameToEdit, const char\* newCountry);

void printCountries(mylist\* head);

void printMenu();

#endif

#include "modul.h"

#include<string>

#include <Windows.h>

#include<iostream>

using namespace std;

mylist\* createList(cherga\*& chergaHead, stack\*& stackHead, int maxYear) {

mylist\* listHead = NULL;

mylist\* listCurrent = NULL;

int numRecords;

cout << "Введіть кількість записів: ";

cin >> numRecords;

for (int i = 0; i < numRecords; ++i) {

mylist\* newListnode = new mylist;

cout << "Введіть ім'я: ";

cin.ignore();

cin.getline(newListnode->data.name, 30);

cout << "Введіть країну: ";

cin.getline(newListnode->data.country, 30);

cout << "Введіть рік: ";

cin >> newListnode->data.year;

cout << "Введіть ціну: ";

cin >> newListnode->data.price;

newListnode->next = NULL;

if (listHead == NULL) {

listHead = newListnode;

}

else {

listCurrent->next = newListnode;

}

listCurrent = newListnode;

if (strcmp(newListnode->data.country, "україна") != 0) {

// Додавання до черги

cherga\* newchergaHead = new cherga;

for (int j = 0; j < 30; ++j) {

newchergaHead->data.name[j] = newListnode->data.name[j];

newchergaHead->data.country[j] = newListnode->data.country[j];

}

newchergaHead->data.year = newListnode->data.year;

newchergaHead->data.price = newListnode->data.price;

newchergaHead->next = NULL;

if (chergaHead == NULL) {

chergaHead = newchergaHead;

}

else {

cherga\* chergaCurrent = chergaHead;

while (chergaCurrent->next != NULL) {

chergaCurrent = chergaCurrent->next;

}

chergaCurrent->next = newchergaHead;

}

}

// додати в стек

if (newListnode->data.year > maxYear) {

stack\* newStackNode = new stack;

for (int j = 0; j < 30; ++j) {

newStackNode->data.name[j] = newListnode->data.name[j];

newStackNode->data.country[j] = newListnode->data.country[j];

}

newStackNode->data.year = newListnode->data.year;

newStackNode->data.price = newListnode->data.price;

newStackNode->next = stackHead;

stackHead = newStackNode;

}

}

return listHead;

}

void printList(mylist\* head) {

cout << "Список студентів:" << endl;

while (head != NULL) {

cout << "Ім'я: " << head->data.name << ", Країна: " << head->data.country

<< ", Рік: " << head->data.year << ", Ціна: " << head->data.price << endl;

head = head->next;

}

}

// Функція для виведення черги на екран

void printQueue(cherga\* chergaHead) {

cout << "Черга студентів:" << endl;

while (chergaHead != NULL) {

cout << "Ім'я: " << chergaHead->data.name << ", Країна: " << chergaHead->data.country

<< ", Рік: " << chergaHead->data.year << ", Ціна: " << chergaHead->data.price << endl;

chergaHead = chergaHead->next;

}

}

// Функція для виведення стеку на екран

void printStack(stack\* stackHead) {

cout << "Стек студентів:" << endl;

while (stackHead != NULL) {

cout << "Ім'я: " << stackHead->data.name << ", Країна: " << stackHead->data.country

<< ", Рік: " << stackHead->data.year << ", Ціна: " << stackHead->data.price << endl;

stackHead = stackHead->next;

}

}

void addToUnorderedList(mylist\*& head, student newData, cherga\*& chergaHead, stack\*& stackHead, int maxYear) {

mylist\* newNode = new mylist;

newNode->data = newData;

newNode->next = head;

head = newNode;

if (strcmp(newData.country, "україна") != 0) {

cherga\* newchergaHead = new cherga;

newchergaHead->data = newData;

newchergaHead->next = NULL;

if (chergaHead == NULL) {

chergaHead = newchergaHead;

}

else {

cherga\* chergaCurrent = chergaHead;

while (chergaCurrent->next != NULL) {

chergaCurrent = chergaCurrent->next;

}

chergaCurrent->next = newchergaHead;

}

}

if (newData.year > maxYear) {

stack\* newStackNode = new stack;

newStackNode->data = newData;

newStackNode->next = stackHead;

stackHead = newStackNode;

}

}

void sortList(mylist\*& head) {

if (head == NULL || head->next == NULL) {

return;

}

bool swapped;

mylist\* ptr;

mylist\* lastPtr = NULL;

do {

swapped = false;

ptr = head;

while (ptr->next != lastPtr) {

if (strcmp(ptr->data.name, ptr->next->data.name) > 0) {

student temp = ptr->data;

ptr->data = ptr->next->data;

ptr->next->data = temp;

swapped = true;

}

ptr = ptr->next;

}

lastPtr = ptr;

} while (swapped);

}

void addToOrderedList(mylist\*& head, student newData) {

mylist\* newNode = new mylist;

newNode->data = newData;

newNode->next = NULL;

if (head == NULL || strcmp(newData.name, head->data.name) < 0) {

newNode->next = head;

head = newNode;

return;

}

mylist\* current = head;

while (current->next != NULL && strcmp(newData.name, current->next->data.name) > 0) {

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

//bool editCountry(mylist\* head, const char\* nameToEdit, const char\* newCountry) {

// mylist\* current = head;

// bool found = false;

//

// while (current != NULL) {

// if (strcmp(current->data.name, nameToEdit) == 0) {

//

// strcpy(current->data.country, newCountry);

// found = true;

// break;

// }

// current = current->next;

// }

//

//

// if (!found) {

// cout << "Студента '" << nameToEdit << "' не знайдено." << endl;

// }

//

// return found;

//}

void printCountries(mylist\* head) {

string countries[100];

int count = 0;

while (head != NULL) {

bool found = false;

for (int i = 0; i < count; ++i) {

if (strcmp(head->data.country, countries[i].c\_str()) == 0) {

found = true;

break;

}

}

if (!found) {

countries[count] = head->data.country;

count++;

}

head = head->next;

}

cout << "Перелік країн:" << endl;

for (int i = 0; i < count; ++i) {

cout << countries[i] << endl;

}

}

void printMenu() {

cout << "Меню вибору опцій:" << endl;

cout << "1. Створення списку, черги та стеку" << endl;

cout << "2. Вивід списку студентів, черги та стеку" << endl;

cout << "3. Додавання студентів" <<endl;

cout << "4. Сортування списку студентів" << endl;

cout << "5. Додавання впорядкованого списку студентів" << endl;

cout << "6. Редагування країни студента" <<endl;

cout << "7. Вивід всіх країн" << endl;

cout << "Ваш вибір (натисніть 13 для виходу): ";

}

**Код основного модуля ( диспетчера – функції main())**

#include <iostream>

#include <Windows.h>

#include <string>

#include "modul.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cherga\* chergaHead = NULL;

stack\* stackHead = NULL;

int k=0; int numStudents;

mylist\* listHead = NULL;

student newStudent;

int maxYear;

char nameToEdit[30];

char newCountry[30];

bool success;

while (k != 13) {

printMenu();

cin >> k;

switch (k)

{

case 1:

cout << "Введіть максимальний рік вступу: ";

cin >> maxYear;

listHead = createList(chergaHead, stackHead, maxYear);

break;

case 2:

printList(listHead);

printQueue(chergaHead);

printStack(stackHead);

break;

case 3:

if (listHead != NULL) {

cout << "Скільки студентів ви хочете додати? ";

cin >> numStudents;

for (int i = 0; i < numStudents; ++i) {

cout << "Введіть дані для студента " << i + 1 << ":" << endl;

cout << "Прізвище: ";

cin >> newStudent.name;

cout << "Країна: ";

cin >> newStudent.country;

cout << "Рік вступу: ";

cin >> newStudent.year;

cout << "Ціна: ";

cin >> newStudent.price;

addToUnorderedList(listHead, newStudent, chergaHead, stackHead, maxYear);

}

}

else {

cout << "Спочатку введіть максимальний рік вступу (вибір 1)." << endl;

}

break;

case 4:

sortList(listHead);

break;

case 5:

if (listHead != NULL) {

cout << "Скільки студентів ви хочете додати? ";

cin >> numStudents;

for (int i = 0; i < numStudents; ++i) {

cout << "Введіть дані для студента " << i + 1 << ":" << endl;

cout << "Прізвище: ";

cin >> newStudent.name;

cout << "Країна: ";

cin >> newStudent.country;

cout << "Рік вступу: ";

cin >> newStudent.year;

cout << "Ціна: ";

cin >> newStudent.price;

addToOrderedList(listHead, newStudent);

}

}

else {

cout << "Спочатку введіть максимальний рік вступу (вибір 1)." << endl;

}

break;

case 6:

/\* cout << "Введіть ім'я: ";

cin >> nameToEdit;

cout << "Введіть країну: ";

cin >> newCountry;

bool success = editCountry(listHead, nameToEdit, newCountry);\*/

break;

case 7:

printCountries(listHead);

break;

default:

break;

}

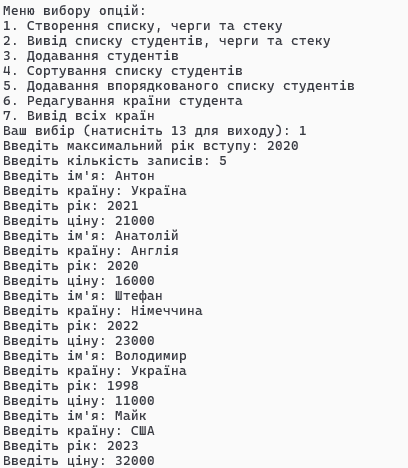
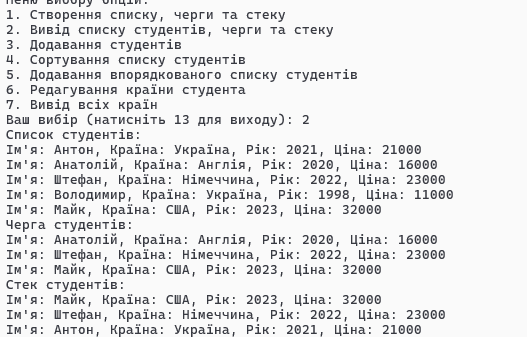
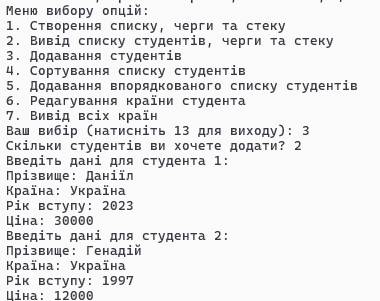
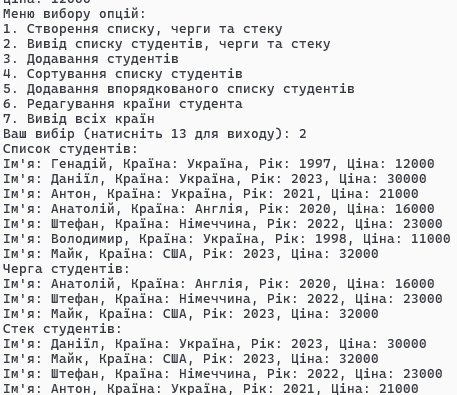
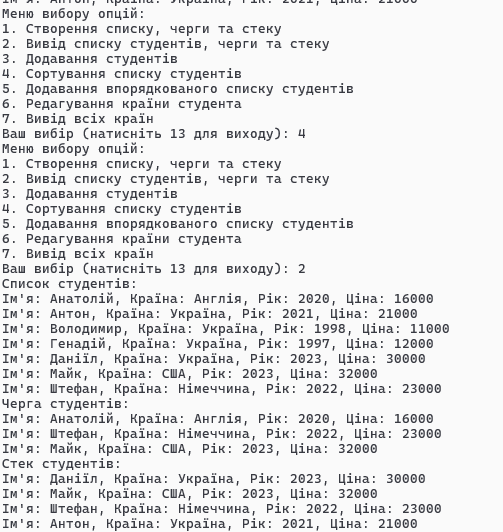
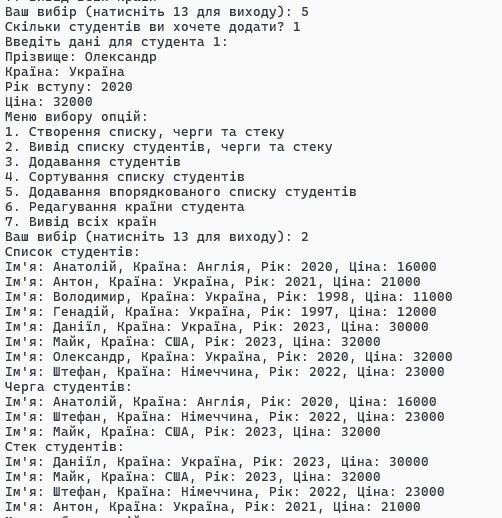
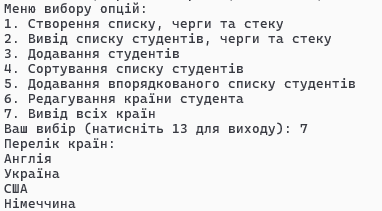
}

}

**Файли проекта:**



**Тестовий набір вхідних даних. Результати роботи усіх функцій проєкту.**

**** ****     

**Висновок:** я набув практичних навиків програмування, тестування програм з використанням стеків та черг.