**Міністерство освіти і науки України**

**Івано-Франківський національний технічний університет нафти і газу**

**Кафедра ІПЗ**

**Лабораторна робота №4**

**Алгоритми для роботи зі чергами**

**Виконав:**

**ст.групи ІП-22-1**

**Чурчак В.В.**

**Перевірив:**

**Григорчук Л.І.**

**Івано-Франківськ**

**2023**

**Мета:** ознайомитися із чергами, реалізувати операції з ними відповідно до варіанту завдання.

**Завдання:**

1. Реалізувати основні функції при роботі з чергами

2. Написати програму для роботи із чергою. Елементом черги зробити власний варіант структури

3. Створити програму яка перевіряє правильність послідовності круглих та квадратних дужок за час, що не перевищує константу множену на довжину цієї послідовності

**Короткі теоретичні відомості**

У C++, черга (queue) є одним із видів контейнерів, який забезпечує упорядковане зберігання елементів та доступ до них за принципом "перший прийшов — перший вийшов" (First In, First Out, або FIFO). Зазвичай це реалізується за допомогою двох операцій: `push` для додавання елемента в кінець черги і `pop` для видалення елемента з початку черги.

Ось приклад використання черги в C++:

Створення об'єкту черги

std::queue<int> myQueue;

Додавання елементів до черги

myQueue.push(10);

myQueue.push(20);

myQueue.push(30);

Отримання значення з початку черги

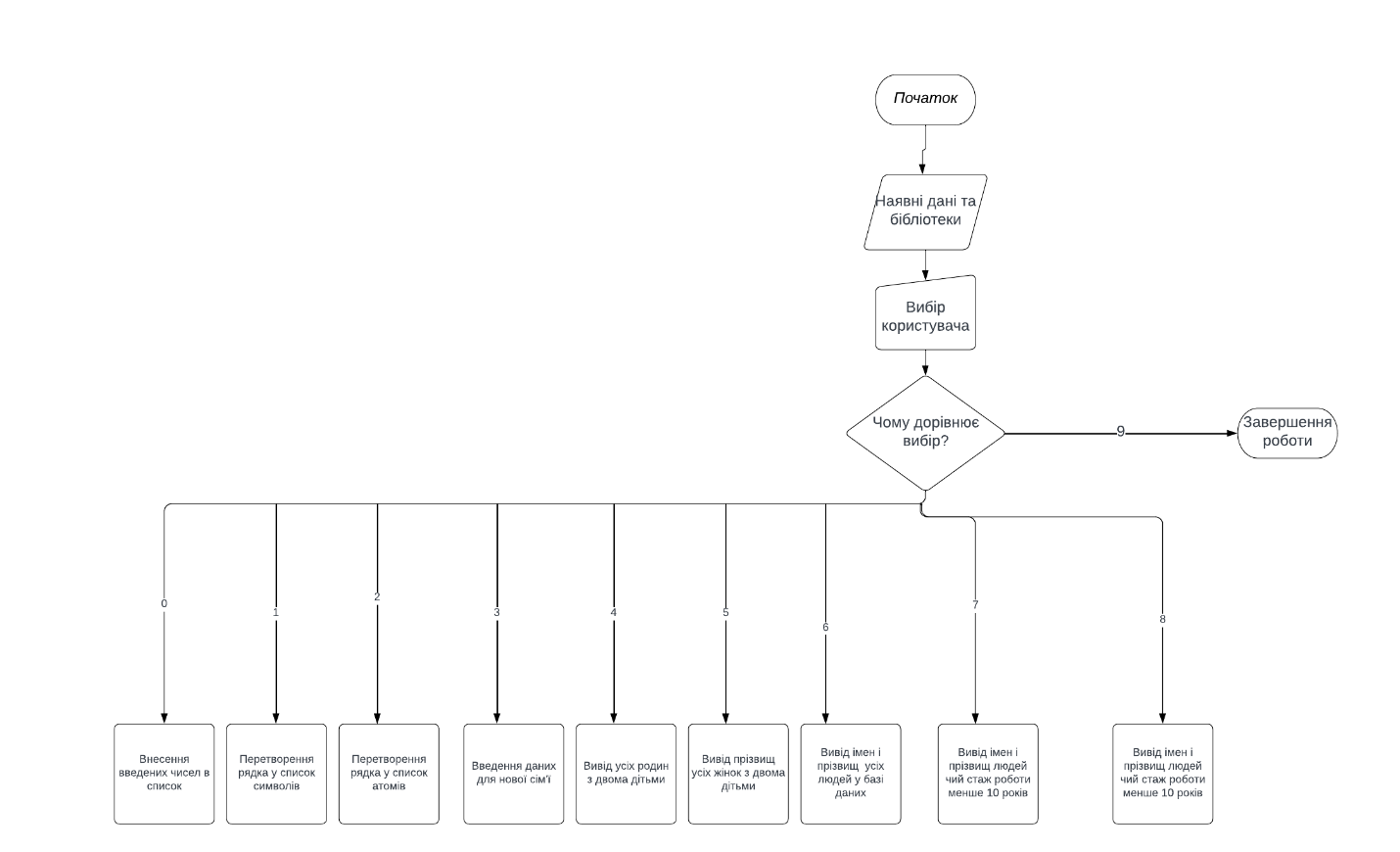
myQueue.front()

Видалення елемента з початку черги

myQueue.pop();

Черги використовуються в різних областях програмування, таких як обробка завдань у порядку їхнього надходження (наприклад, в чергах подій), реалізація алгоритмів обходу графів та багато інших випадків, де важливий порядок обробки даних.

**Графічний алгоритм**



**Виконання**

#include <iostream>

#include <string>

#include <queue>

#include <Windows.h>

using namespace std;

int numOfChildren=0;

struct Date {

int day;

int month;

int year;

Date() : day(1), month(1), year(1900) {}

};

struct Job {

bool employed = true;

string profession;

int experience = 0;

};

struct Person {

string firstName;

string lastName;

Date birthDate;

Job jobInfo;

};

struct Family {

bool hasChildren = true;

Person husband;

Person wife;

vector<Person> children;

};

queue<Family> families;

void getAllHumanData() {

// Виводимо імена і прізвища всіх людей з БД

queue<Family> tempQueue;

cout << "Всі люди в базі даних:\n";

while (!families.empty()) {

const Family& family = families.front();

cout << "Чоловік: " << family.husband.firstName << " " << family.husband.lastName << endl;

cout << "Дата народження: " << family.husband.birthDate.day << " " << family.husband.birthDate.month << " " << family.husband.birthDate.year << endl;

if (family.husband.jobInfo.employed==true) {

cout << "Робота є: " <<endl;

cout << "Назва роботи: " << family.husband.jobInfo.profession << endl;

cout << "Кількість років досвіду " << family.husband.jobInfo.profession << endl;

}

else {

cout << "Роботи немає: " << endl;

}

cout << "Дружина: " << family.wife.firstName << " " << family.wife.lastName << std::endl;

cout << "Дата народження: " << family.wife.birthDate.day << " " << family.wife.birthDate.month << " " << family.wife.birthDate.year << endl;

if (family.wife.jobInfo.employed == true) {

cout << "Робота є: " << endl;

cout << "Назва роботи: " << family.wife.jobInfo.profession << endl;

cout << "Кількість років досвіду " << family.wife.jobInfo.profession << endl;

}

else {

cout << "Роботи немає: " << endl;

}

if (family.hasChildren == true) {

for (const Person& child : family.children) {

cout << "Дитина: " << child.firstName << " " << child.lastName << std::endl;

cout << "Дата народження: " << child.birthDate.day << " " << child.birthDate.month << " " << child.birthDate.year << endl;

}

}

// Зберігаємо видалений елемент у тимчасову чергу

tempQueue.push(families.front());

families.pop(); // Видаляємо перший елемент з черги

}

// Повертаємо всі видалені елементи назад у основну чергу

while (!tempQueue.empty()) {

families.push(tempQueue.front());

tempQueue.pop();

}

}

void addFamilyMember() {

Family currentFamily;

cout << "Ім'я чоловіка: ";

cin >> currentFamily.husband.firstName;

cout << "Прізвище чоловіка: ";

cin >> currentFamily.husband.lastName;

cout << "Місяць народження: ";

cin >> currentFamily.husband.birthDate.month;

cout << "День народження: ";

cin >> currentFamily.husband.birthDate.day;

cout << "Рік народження: ";

cin >> currentFamily.husband.birthDate.year;

string isWorking;

bool checker = false;

while (checker != true) {

cout << "Має роботу? (y/n)";

cin >> isWorking;

if (isWorking == "y") {

currentFamily.husband.jobInfo.employed = true;

checker = true;

}

else if (isWorking == "n") {

currentFamily.husband.jobInfo.employed = false;

checker = true;

}

}

if (currentFamily.husband.jobInfo.employed == true) {

cout << "Назва професії: ";

cin >> currentFamily.husband.jobInfo.profession;

cout << "Кількість років досвіду: ";

cin >> currentFamily.husband.jobInfo.experience;

}

cout << "Ім'я дружини: ";

cin >> currentFamily.wife.firstName;

cout << "Прізвище дружини: ";

cin >> currentFamily.wife.lastName;

cout << "Місяць народження: ";

cin >> currentFamily.wife.birthDate.month;

cout << "День народження: ";

cin >> currentFamily.wife.birthDate.day;

cout << "Рік народження: ";

cin >> currentFamily.wife.birthDate.year;

isWorking = "a";

checker = false;

while (checker == false) {

cout << "Має роботу? (y/n)";

cin >> isWorking;

if (isWorking == "y") {

currentFamily.wife.jobInfo.employed = true;

checker = true;

}

else if (isWorking == "n") {

currentFamily.wife.jobInfo.employed = false;

checker = true;

}

}

if (currentFamily.wife.jobInfo.employed == true) {

cout << "Назва професії: ";

cin >> currentFamily.wife.jobInfo.profession;

cout << "Кількість років досвіду: ";

cin >> currentFamily.wife.jobInfo.experience;

}

cout << "Скільки дітей у цій сім'ї? ";

cin >> numOfChildren;

if (numOfChildren > 0) {

currentFamily.hasChildren = true;

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

Person child;

cout << "Ім'я дитини " << i + 1 << ": ";

cin >> child.firstName;

cout << "Прізвище дитини " << i + 1 << ": ";

cin >> child.lastName;

cout << "Місяць народження: ";

cin >> child.birthDate.month;

cout << "День народження: ";

cin >> child.birthDate.day;

cout << "Рік народження: ";

cin >> child.birthDate.year;

currentFamily.children.push\_back(child);

}

}

else {

currentFamily.hasChildren = false;

}

families.push(currentFamily);

}

void clearAll(queue<Family>& myQueue) {

std::queue<Family> emptyQueue;

while (!myQueue.empty()) {

myQueue.pop();

}

cout << "Дані успішно очищено\n";

}

void remove\_elements(queue<Family>& myQueue) {

if (myQueue.empty()) {

cout << "Черга порожня\n";

}

else {

cout << "Сім'ї без дітей успішно очищені\n";

// Створення допоміжної черги для збереження елементів, які не потрібно видаляти

queue<Family> tempQueue;

Family currentFamily = myQueue.front();

// Перебір елементів у вихідній черзі

while (!myQueue.empty()) {

Family currentFamily = myQueue.front();

myQueue.pop();

// Умова для збереження елемента в допоміжній черзі (не видаляти)

if (currentFamily.hasChildren == true) {

tempQueue.push(currentFamily);

}

}

// Перенесення елементів з допоміжної черги назад в оригінальну

myQueue = tempQueue;

}

}

int choice = 0;

int main() {

// Створюємо родину

Family family;

// Додаємо інформацію про чоловіка

family.husband.firstName = "Іван";

family.husband.lastName = "Петров";

family.husband.birthDate.day = 10;

family.husband.birthDate.month = 5;

family.husband.birthDate.year = 1980;

family.husband.jobInfo.employed = true;

family.husband.jobInfo.profession = "інженер";

family.husband.jobInfo.experience = 10;

// Додаємо інформацію про дружину

family.wife.firstName = "Марія";

family.wife.lastName = "Петрова";

family.wife.birthDate.day = 15;

family.wife.birthDate.month = 8;

family.wife.birthDate.year = 1982;

family.wife.jobInfo.employed = true;

family.wife.jobInfo.profession = "вчитель";

family.wife.jobInfo.experience = 8;

family.hasChildren = true;

// Додаємо інформацію про дітей

Person child1;

child1.firstName = "Анна";

child1.lastName = "Петрова";

child1.birthDate.day = 5;

child1.birthDate.month = 2;

child1.birthDate.year = 2005;

Person child2;

child2.firstName = "Петро";

child2.lastName = "Петров";

child2.birthDate.day = 20;

child2.birthDate.month = 12;

child2.birthDate.year = 2008;

family.children.push\_back(child1);

family.children.push\_back(child2);

families.push(family);

Family family1;

// Додаємо інформацію про чоловіка

family1.husband.firstName = "Василь";

family1.husband.lastName = "Іванов";

family1.husband.birthDate.day = 23;

family1.husband.birthDate.month = 1;

family1.husband.birthDate.year = 1990;

family1.husband.jobInfo.employed = false;

// Додаємо інформацію про дружину

family1.wife.firstName = "Катерина";

family1.wife.lastName = "Іванова";

family1.wife.birthDate.day = 21;

family1.wife.birthDate.month = 10;

family1.wife.birthDate.year = 1972;

family1.wife.jobInfo.employed = true;

family1.wife.jobInfo.profession = "вчитель";

family1.wife.jobInfo.experience = 15;

family1.hasChildren = true;

// Додаємо інформацію про дітей

Person child3;

child3.firstName = "Галина";

child3.lastName = "Іванова";

child3.birthDate.day = 5;

child3.birthDate.month = 2;

child3.birthDate.year = 2006;

Person child4;

child4.firstName = "Кирило";

child4.lastName = "Іванов";

child4.birthDate.day = 20;

child4.birthDate.month = 2;

child4.birthDate.year = 2007;

Person child5;

child5.firstName = "Георгій";

child5.lastName = "Іванов";

child5.birthDate.day = 12;

child5.birthDate.month = 7;

child5.birthDate.year = 2008;

family1.children.push\_back(child3);

family1.children.push\_back(child4);

family1.children.push\_back(child5);

families.push(family1);

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

do {

cout << "1. Ввести дані для нової сім'ї\n";

cout << "2. Вивести імена і прізвища всіх людей в базі даних\n";

cout << "3. Очистити усі дані\n";

cout << "4. Видалити усі сім'ї без дітей\n";

cout << "9. Вихід\n";

std::cin >> choice;

switch (choice) {

case 1:

addFamilyMember();

break;

case 2:

getAllHumanData();

break;

case 3:

clearAll(families);

break;

case 4:

remove\_elements(families);

break;

default:

break;

}

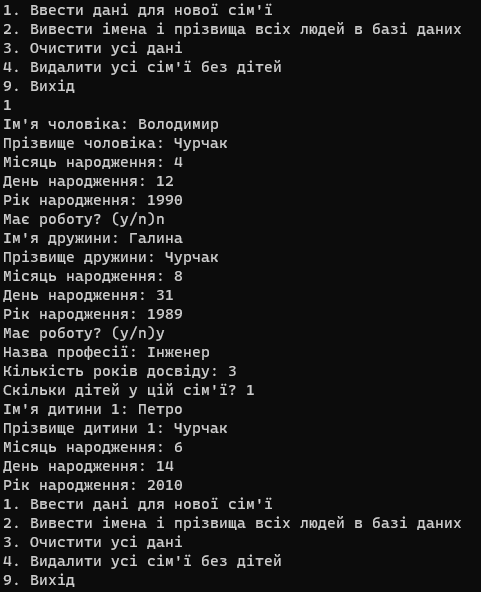
} while (choice != 5);

return 0;

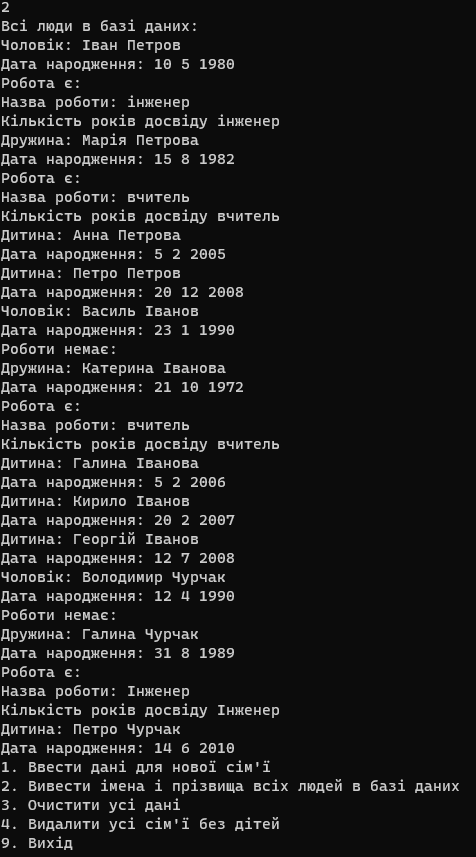
}

**Результати:**

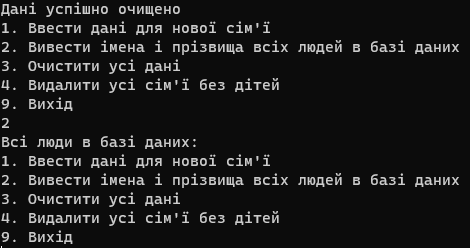
1. **Ввід нової сім’ї в чергу**



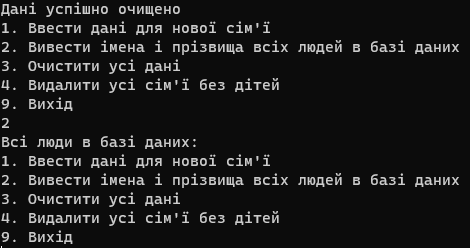
1. **Вивід усіх сімей у черзі**



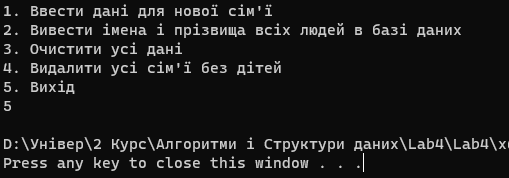
1. **Очищення сімей без дітей**



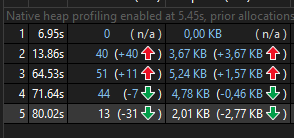
1. **Очищення усіх даних в черзі**



1. **Вихід з програми**



**Визначення затрат пам’яті на різних етапах виконання програми**



Затрати пам’яті значно збільшуються при заповненні черги але при очищенні вони зменшуються. Отже програма добре оптимізована

**Висновок:** під час виконання лабораторної роботи я ознайомився з теоретичними відомостями і виконав завдання відповідно до варіанту. Була реалізована робота з чергами: їх заповнення, вивід елементів зі черг, видалення та методи копіювання.