Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Звіт

з лабораторної роботи №4

«Особливі види черг»

з дисципліни «Алгоритми та структури даних»

Виконала:

ст.гр. ІП-22-4

Томин Є.В.

Перевірив:

Григорчук Л. І.

Івано-Франківськ

2023

**Тема.** Особливі види черг

**Мета:** набуття практичних вмінь та навичок при організації з особливими видами черг – деками та чергами з пріорітетами.

**Теоретичні відомості**

**Черга, як і стек** - це один із різновидів однозв’язного лінійного списку. Вона працює за принципом «першим прийшов – першим вийшов», що позначається абревіатурою FIFO (від англ. First In First Out), і характеризується такими властивостями:

⮚Елементи додаються в кінець черги;

⮚Елементи зчитуються та видаляються з початку (вершини) черги;

⮚Покажчик в останньому елементі черги дорівнює nil;

⮚Неможливо отримати елемент із середини черги, не вилучивши всі елементи, що йдуть попереду.

**Черга з пріоритетами** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *priority queue*) — це [структура даних](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), що призначена для обслуговування множини елементів, кожний з яких додатково має "пріоритет", пов'язаний з ним. У пріоритетній черзі першим обслуговується елемент, який має найвищий пріоритет, відповідно елемент, що має найнижчий пріоритет буде обслугований останнім. У деяких реалізаціях, якщо два елементи мають однаковий пріоритет, вони подаються відповідно до порядку, в якому вони були закладені, в той час як в інших реалізаціях упорядкування елементів з однаковим пріоритетом не визначено.

**Дек** - особливий вид черги. Дек (від англ. Deq - double ended queue, тобто черга з двома кінцями) - це такий послідовний список, в якому як включення, так і виключення елементів може здійснюватися з будь-якого з двох кінців списку. Окремий випадок дека - дек з обмеженим входом і дек з обмеженим виходом. Логічна і фізична структури дека аналогічні логічної і фізичної структурі кільцевої FIFO-черзі. Однак, стосовно до деку доцільно говорити не про початок і кінець, а про лівий і правий кінці.

**Операції над Деком:**

•включення елемента справа;

•включення елемента зліва;

•виключення елемента справа;

•виключення елемента зліва;

•визначення розміру;

•очищення.

**8.3 Завдання для самостійного виконання**

**1.**Реалізувати основні функції по роботі з деками, в яких записані цілочисельні

дані. Ввести функцію контролю (виводу) поточного стану дека.

**2.** Перевірити, чи введений рядок є паліндромом.

**3.** Написати програму для роботи з чергою із пріоритетним включенням.

Елементом черги зробити родину().

Пріоритетними вважати:

- кількість дітей(число)

- кількість доходів (число)

- Інформація про роботу (працює/не працює)

-Вік членів родини (у роках)

**Зауваження І.** Додавання елементу до такої черги відрізняється від

додавання елементу до звичайної черги необхідністю пошуку відповідного місця

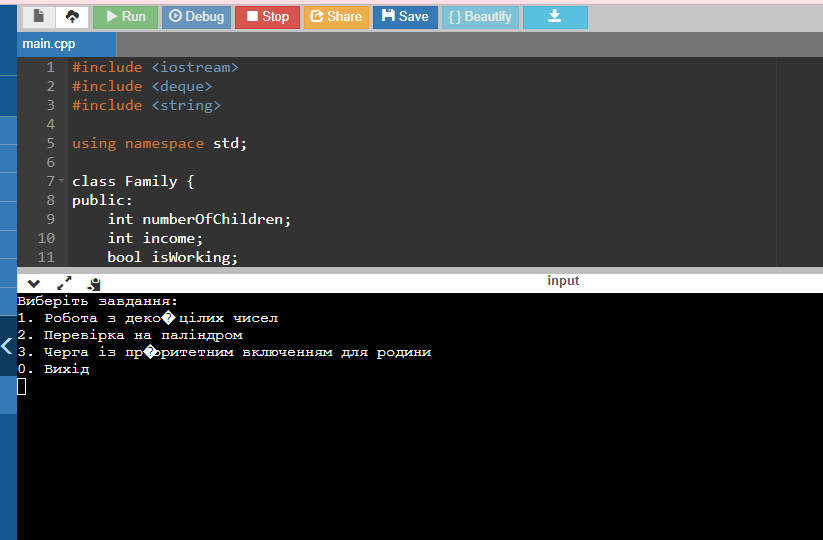
в черзі

**Зауваження 2.** Чергу можна реалізувати або на основі двозвʼязного списку,

або на основі кільцевого однозвʼязного списку.

**Виконання**

**Меню завдань**

****

**Завдання 1**

#include <iostream>

#include <deque>

#include <stdexcept>

using namespace std;

// Функція для виведення поточного стану дека

void printDeque(const deque<int>& myDeque) {

cout << "Дек: ";

for (int element : myDeque) {

cout << element << " ";

}

cout << endl;

}

int main() {

deque<int> myDeque;

cout << "Введіть елементи для додавання до деку (введіть 'k', щоб завершити введення):" << endl;

char userInput;

while (true) {

cin >> userInput;

if (userInput == '0' || userInput == 'k') {

break;

}

myDeque.push\_back(userInput - '0'); // Введене 'k' не буде додаватися до деку

}

// Вивід поточного стану деку викликаючи функцію

printDeque(myDeque);

// Перевірка, чи дек є паліндромом

bool isPalindrome = true;

for (int i = 0; i < myDeque.size() / 2; i++) {

if (myDeque[i] != myDeque[myDeque.size() - 1 - i]) {

isPalindrome = false;

break;

}

}

if (isPalindrome) {

cout << "Дек є паліндромом." << endl;

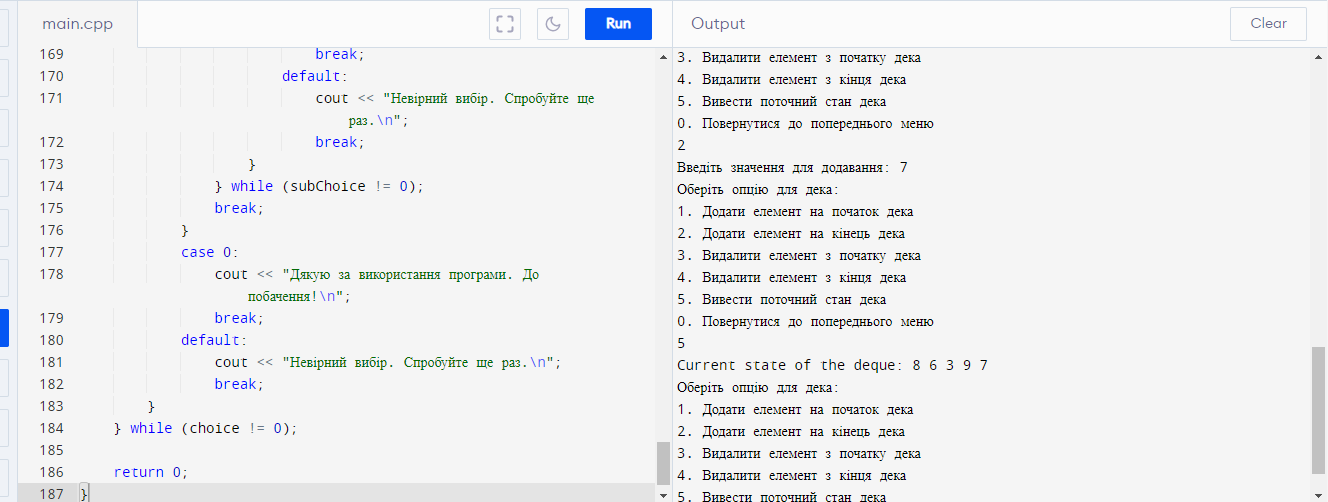
} else {

cout << "Дек не є паліндромом." << endl;

}

return 0;

}



**Мал.1.** Записування цілочисельних даних у дек. Вивід функції контролю (виводу) поточного стану дека.

**Завдання 2**

#include <iostream>

#include <deque>

#include <stdexcept>

using namespace std;

// Функція для виведення поточного стану дека

void printDeque(const deque<int>& myDeque) {

cout << "Дек: ";

for (int element : myDeque) {

cout << element << " ";

}

cout << endl;

}

int main() {

deque<int> myDeque;

cout << "Введіть елементи для додавання до деку (введіть 'k', щоб завершити введення):" << endl;

char userInput;

while (true) {

cin >> userInput;

if (userInput == '0' || userInput == 'k') {

break;

}

myDeque.push\_back(userInput - '0'); // Введене 'k' не буде додаватися до деку

}

// Вивід поточного стану деку викликаючи функцію

printDeque(myDeque);

// Перевірка, чи дек є паліндромом

bool isPalindrome = true;

for (int i = 0; i < myDeque.size() / 2; i++) {

if (myDeque[i] != myDeque[myDeque.size() - 1 - i]) {

isPalindrome = false;

break;

}

}

if (isPalindrome) {

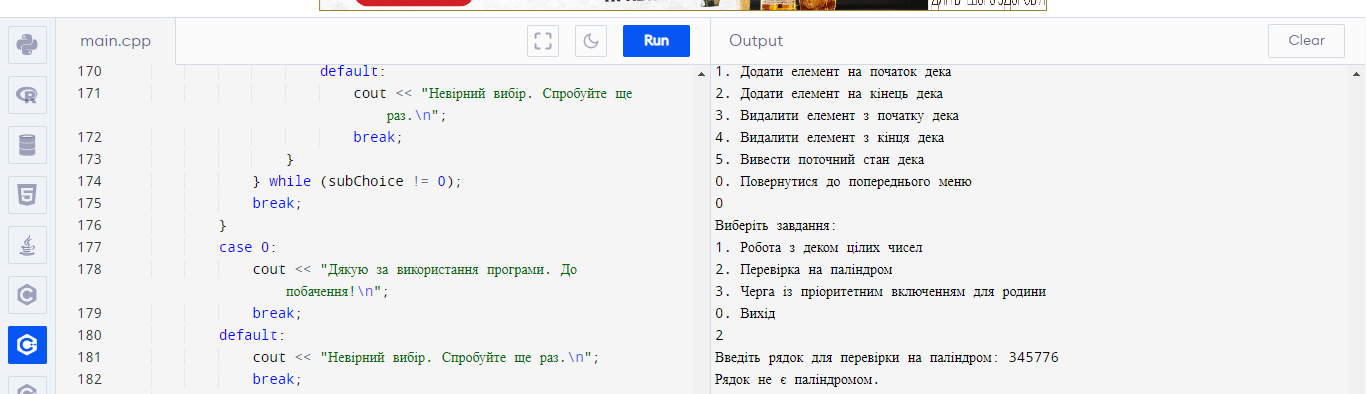
cout << "Дек є паліндромом." << endl;

} else {

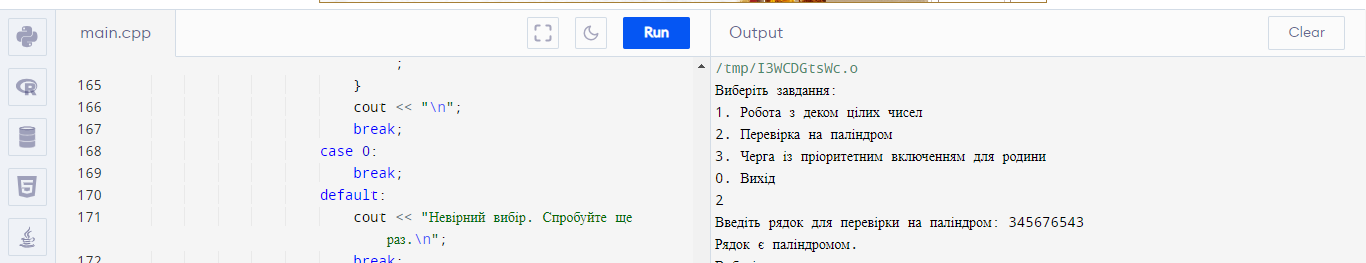
cout << "Дек не є паліндромом." << endl;

}

return 0;



**Мал.2.1.** Перевірка, чи введений рядок є паліндромом (дек не є паліндромом).



**Мал.2.2.** Перевірка, чи введений рядок є паліндромом (дек є паліндромом).

**Завдання 3**

#include <iostream>

#include <string>

#include <vector>

#include <queue>

using namespace std;

// Структура день народження

struct DateOfBirth {

int day;

int month;

int year;

};

// Структура інформація про роботу

struct JobInfo {

bool notWorking;

int jobCode;

double salary;

};

// Структура членів сім'ї

struct FamilyMember {

string firstName;

string lastName;

DateOfBirth dob;

JobInfo job;

};

// Структура про сім'ю

struct Family {

FamilyMember husband;

FamilyMember wife;

vector<FamilyMember> children;

// Функція для обчислення пріоритету родини

int getPriority() const {

int priority = 0;

priority += children.size(); // Пріоритет за кількістю дітей

priority += husband.job.salary + wife.job.salary; // Пріоритет за доходами

priority += (husband.job.notWorking ? 0 : 1); // Пріоритет за роботою

priority += (wife.job.notWorking ? 0 : 1);

priority += (2023 - husband.dob.year); // Пріоритет за віком

priority += (2023 - wife.dob.year);

return priority;

}

// Перевизначення оператора менше для порівняння за пріоритетом

bool operator<(const Family& other) const {

return getPriority() < other.getPriority();

}

};

int main() {

priority\_queue<Family> familyQueue;

// Додавання сімей у чергу

Family family1{

{"Назар", "Будніков", {1, 1, 1980}, {false, 101, 50000}},

{"Євгенія", "Буднікова", {15, 5, 1982}, {false, 102, 45000}},

{

{"Петро", "Будніков", {10, 2, 2005}, {false, 0, 0}},

{"Діма", "Будніков", {20, 4, 2007}, {false, 0, 0}},

{"Ваня", "Будніков", {5, 9, 2010}, {false, 0, 0}}

}

};

familyQueue.push(family1);

Family family2{

{"Дмитро", "Чорній", {2, 3, 1979}, {false, 104, 70000}},

{"Наталія", "Чорній", {19, 8, 1981}, {false, 115, 40000}},

{

{"Оля", "Чорній", {10, 2, 2003}, {false, 0, 0}},

{"Настя", "Чорній", {10, 2, 2003}, {false, 0, 0}}

}

};

familyQueue.push(family2);

// Виведення сімей із черги у порядку пріоритету

while (!familyQueue.empty()) {

Family topFamily = familyQueue.top();

familyQueue.pop();

// Виведення інформації про сім'ю

cout << "Husband: " << topFamily.husband.firstName << " " << topFamily.husband.lastName << endl;

cout << "Wife: " << topFamily.wife.firstName << " " << topFamily.wife.lastName << endl;

cout << "Children: " << topFamily.children.size() << endl;

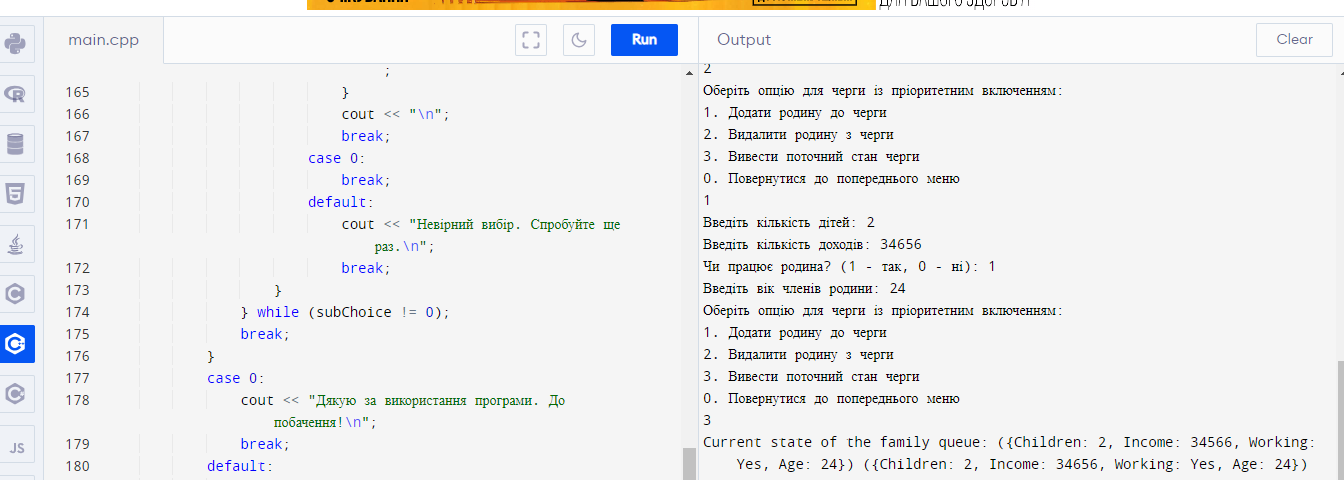
cout << "Priority: " << topFamily.getPriority() << endl;

cout << "------------------------" << endl;

}

return 0;

}

****

**Мал.3.** Програма для роботи з чергою із пріоритетним включенням. Виконана на основі кільцевого однозв'язного списку, реалізованого в стандартній бібліотеці C++ через priority\_queue, для реалізації черги із пріоритетним включенням.

**Загальний код програми**

#include <iostream>

#include <deque>

#include <string>

using namespace std;

class Family {

public:

int numberOfChildren;

int income;

bool isWorking;

int age;

// Конструктор класу

Family(int children, int money, bool working, int familyAge)

: numberOfChildren(children), income(money), isWorking(working), age(familyAge) {}

};

void printDeque(const deque<int>& myDeque) {

cout << "Current state of the deque: ";

for (int value : myDeque) {

cout << value << " ";

}

cout << "\n";

}

bool isPalindrome(const string& str) {

int start = 0;

int end = str.length() - 1;

while (start < end) {

if (str[start] != str[end]) {

return false;

}

start++;

end--;

}

return true;

}

int main() {

deque<int> myDeque; // Дек для завдання 1

int choice;

do {

cout << "Виберіть завдання:\n"

"1. Робота з деком цілих чисел\n"

"2. Перевірка на паліндром\n"

"3. Черга із пріоритетним включенням для родини\n"

"0. Вихід\n";

cin >> choice;

switch (choice) {

case 1: {

deque<int> myDeque; // Дек для завдання 1

int subChoice;

do {

cout << "Оберіть опцію для дека:\n"

"1. Додати елемент на початок дека\n"

"2. Додати елемент на кінець дека\n"

"3. Видалити елемент з початку дека\n"

"4. Видалити елемент з кінця дека\n"

"5. Вивести поточний стан дека\n"

"0. Повернутися до попереднього меню\n";

cin >> subChoice;

switch (subChoice) {

case 1: {

int value;

cout << "Введіть значення для додавання: ";

cin >> value;

myDeque.push\_front(value);

break;

}

case 2: {

int value;

cout << "Введіть значення для додавання: ";

cin >> value;

myDeque.push\_back(value);

break;

}

case 3:

if (!myDeque.empty()) {

myDeque.pop\_front();

} else {

cout << "Дек порожній.\n";

}

break;

case 4:

if (!myDeque.empty()) {

myDeque.pop\_back();

} else {

cout << "Дек порожній.\n";

}

break;

case 5:

printDeque(myDeque);

break;

case 0:

break;

default:

cout << "Невірний вибір. Спробуйте ще раз.\n";

break;

}

} while (subChoice != 0);

break;

}

case 2: {

string str;

cout << "Введіть рядок для перевірки на паліндром: ";

cin >> str;

if (isPalindrome(str)) {

cout << "Рядок є паліндромом.\n";

} else {

cout << "Рядок не є паліндромом.\n";

}

break;

}

case 3: {

deque<Family> familyQueue; // Черга для завдання 3

int subChoice;

do {

cout << "Оберіть опцію для черги із пріоритетним включенням:\n"

"1. Додати родину до черги\n"

"2. Видалити родину з черги\n"

"3. Вивести поточний стан черги\n"

"0. Повернутися до попереднього меню\n";

cin >> subChoice;

switch (subChoice) {

case 1: {

int children , money, age;

bool working;

cout << "Введіть кількість дітей: ";

cin >> children;

cout << "Введіть кількість доходів: ";

cin >> money;

cout << "Чи працює родина? (1 - так, 0 - ні): ";

cin >> working;

cout << "Введіть вік членів родини: ";

cin >> age;

Family newFamily(children, money, working, age);

familyQueue.push\_back(newFamily);

break;

}

case 2:

if (!familyQueue.empty()) {

familyQueue.pop\_front();

} else {

cout << "Черга порожня.\n";

}

break;

case 3:

cout << "Current state of the family queue: ";

for (const Family& f : familyQueue) {

cout << "({Children: " << f.numberOfChildren

<< ", Income: " << f.income

<< ", Working: " << (f.isWorking ? "Yes" : "No")

<< ", Age: " << f.age << "}) ";

}

cout << "\n";

break;

case 0:

break;

default:

cout << "Невірний вибір. Спробуйте ще раз.\n";

break;

}

} while (subChoice != 0);

break;

}

case 0:

cout << "Дякую за використання програми. До побачення!\n";

break;

default:

cout << "Невірний вибір. Спробуйте ще раз.\n";

break;

}

} while (choice != 0);

return 0;

}

**Висновок.**

Виконуючи лабораторну роботу, я навчилася працювати з особливими видами черг – деками та чергами з пріорітетами. Навчилася створювати програму на основі кільцевого однозв’язного списку.