ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О.ГОНЧАРА ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

# ЗВІТ

# З ЛАБОРАТОРНОЇ РОБОТИ №1

# за курсом «Програмування»

# Варіант № 5

Виконав

студент групи МА-21-1

Гудзь Вячеслав

Дніпро, 2023

1. Постановка задачі

2. Опис розв’язку

3. Вихідний код програми

4. Опис інтерфейсу програми(робота з користувачем)

5. Аналіз завдання

Постановка задачі

Розробити об’єктно-орієнтовану бібліотеку для роботи зі структурами даних за однією з нижченаведених тем у відповідності з нижченаведеними вимогами.

Властивості та методи для класів розробити у відповідності з відомими визначеннями відповідних структур даних.

Скласти тести для перевірки працездатності бібліотеки. Скласти програму, що демонструє можливості розробленої бібліотеки.

Загальні вимоги

В незалежності від індивідуального варіанта повинні бути реалізовані наступні можливості:

1. Реалізація методів ініціалізації (конструктор за замовчуванням та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []), присвоювання (перевантаження =), візуалізації, збереження (на диск) та відновлення, діалогового керування, "розумного доступу" (перевантаження ->), а також псевдо змінних (забезпечення можливості виду: f(x)=const).

2. Перевантаження (спільне використання) потокового введення/виведення. (введення з файл, виведення в файл)

3. Створення та використання файла бібліотеки (\*.lib, або \*.obj).

4. Повторне використання класів без їх перекомпіляції (“ReUse”, тобто наслідування).

5. Застосування вказаної структури даних для розв’язання типової задачі. Результати виконання лабораторної роботи повинні бути викладені у вигляді звіту.

В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому режимі (за допомогою меню) виконувати наступні операції:

1. Створювати об’єкти класу (у довільній кількості).

2. Вводити/виводити об’єкти на диск.

3. Візуалізувати об’єкти.

4. Виконувати операції над об’єктами.

Теми індивідуальних завдань

Черги та пріоритети. При додаванні елемента до черги визначається пріоритет його обслуговування. При видаленні елементу з черги обирається елемент з більшим пріоритетом. Використання псевдозмінних для черги аналогічно використанню псевдозмінних для стека

# Опис розв’язку

Завдання було виконано за допомогою основних інструментів Visual Studio 2022, в якій можна зробити статичну бібліотеку і використовувати в одному розв’язку два проекти. Вимоги були виконані і показані у реалізації.

Моїм варіантом був варіант 3, а саме реалізація черги з її основними методами. Додатково була реалізація псевдозмінної та робота з різними типами даних та файлів.

Вихідний код програми

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <algorithm>

#include <iterator>

#include <sstream>

using namespace std;

struct people

{

int priority = 0;

std::string name = "no\_name";

};

class que

{

people\* arr;

int capacity;

int rear;

int count;

int left = -1;

int right = -1;

public:

que(int size = 200);

que(const que& q);

~que();

void pop();

void push(people x);

void incapacity(int newcapacity = 10);

void show();

void file(std::string filename);

void setpriority();

people peek();

int size();

bool isEmpty();

bool isFull();

que& getfromfile(std::string filename);

que& operator+(const que& q);

que& operator=(const people& p);

que& operator = (const que& q);

que& operator ()(int l, int r);

friend ostream& operator <<(std::ostream& os, const que& q);

friend istream& operator >>(std::istream& is, que& q);

};

#include "My\_queue.h"

using namespace std;

que::que(int size)

{

arr = new people[size];

capacity = size;

rear = -1;

count = 0;

}

que::que(const que& q)

{

this->arr = new people[q.capacity];

capacity = q.capacity;

rear = -1;

count = 0;

for (int i = 0; i < q.count; i++)

{

push(q.arr[i]);

}

}

que::~que() {

delete[] arr;

}

void que::pop() //удаления переднего элемента из очереди

{

people\* temp = new people[this->count - 1];

for (int i = 1; i < this->count; i++)

{

temp[i - 1] = this->arr[i];

}

delete[] this->arr;

this->arr = new people[this->capacity];

this->count--;

for (int i = 0; i < this->count; i++)

{

this->arr[i] = temp[i];

}

}

void que::push(people item) //добавления элемента в queue

{

if (isFull())

{

incapacity();

}

rear = (rear + 1) % capacity;

arr[rear] = item;

count++;

this->setpriority();

}

people que::peek() //возврат первого элемента queue

{

if (isEmpty())

{

cout << "Underflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

return arr[0];

}

int que::size() {

return count;

}

bool que::isEmpty() {

return (size() == 0);

}

bool que::isFull() {

return (size() == capacity);

}

que& que::operator=(const people& p)

{

if (this->left >= 0 && this->right <= count - 1)

{

for (int i = this->left - 1; i < this->right; i++)

{

this->arr[i] = p;

}

}

return \*this;

}

que& que::operator=(const que& q)

{

if (this->left >= 0 && this->right <= count - 1)

{

int i = this->left - 1, j = q.left;

for (i, j; i < this->right; i++, j++)

{

this->arr[i] = q.arr[j];

}

return \*this;

}

if (this->arr != nullptr)

{

delete[] this->arr;

}

this->arr = new people[q.capacity];

capacity = q.capacity;

rear = -1;

count = 0;

for (int i = 0; i < q.count; i++)

{

push(q.arr[i]);

}

return \*this;

}

que& que::operator()(int l, int r)

{

if (r > count || l < 0)

{

cout << "Неправильные границы\n";

system("pause");

return \*this;

}

this->left = l;

this->right = r;

return \*this;

}

void que::incapacity(int newcapacity) //увеличение capacity

{

if (newcapacity >= capacity) { capacity = newcapacity; }

else

{

capacity += newcapacity;

}

people\* temp = new people[this->capacity];

for (int i = 0; i < this->count; i++)

{

temp[i] = this->arr[i];

}

delete[] this->arr;

arr = new people[capacity];

for (int i = 0; i < this->count; i++)

{

this->arr[i] = temp[i];

}

delete[] temp;

}

void que::show()

{

cout << "\tИмя\t\tПриоритет" << endl;

for (int i = 0; i < this->count; i++)

{

cout << i + 1 << ". " << arr[i].name << "\t\t " << arr[i].priority << endl;

}

cout << endl;

}

void que::file(std::string filename)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::ofstream file;

file.open(filename);

file << \*this;

file.close();

SetConsoleCP(866);

SetConsoleOutputCP(866);

}

que& que::getfromfile(std::string filename)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::fstream file;

file.open(filename);

while (!file.is\_open())

{

std::cout << "Не верный путь к файлу введите снова\n";

char path[100];

std::cin >> path;

file.open(path);

}

file >> \*this;

file.close();

SetConsoleCP(866);

SetConsoleOutputCP(866);

return \*this;

}

void que::setpriority()

{

people temp;

for (int i = 0; i < this->count - 1; i++)

{

for (int j = 0; j < this->count - i - 1; j++)

{

if(this->arr[j].priority > this->arr[j + 1].priority)

{

temp = this->arr[j];

this->arr[j] = this->arr[j + 1];

this->arr[j + 1] = temp;

}

}

}

}

que& que::operator+(const que& q)

{

for (int i = 0; i < q.count; i++)

{

this->push(q.arr[i]);

}

return \*this;

}

ostream& operator<<(ostream& os, const que& q)

{

for (int i = 0; i < q.count; i++)

{

os << i + 1 << ". \"" << q.arr[i].name << "\"\t\t" << q.arr[i].priority << endl;

}

return os;

}

istream& operator >>(istream& is, que& q)

{

people p;

stringstream ss;

ss << is.rdbuf();

for (int i = 5; i < ss.str().size(); i++)

{

if (ss.str()[i] == '\"')

{

int j = i + 1, x = 0;

string tempname, temp;

while (ss.str()[j] != '\"')

{

tempname += ss.str()[j];

j++;

}

j += 2;

x = ss.str().find('\n', j);

if (x != std::string::npos)

{

temp = ss.str().substr(j, x);

}

else

{

temp = ss.str().substr(j, 1);

}

p.name = tempname;

p.priority = stoi(temp);

i = j + 2;

q.push(p);

}

}

return is;

}

#include <D:\Projects\Static\_library\My\_queue\_library\My\_queue.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

que\* qarr = new que[10];

people p;

string filename = "", name = "no\_name";

int priority = 1, l = 0, r = 0, i = 0, l1 = 0, r1 = 0;

int nfun = 0, nq = 0;

while (nfun != -1)

{

cout << "0 = Записать данные в файл\n"

<< "1 = Показать очередь\n"

<< "2 = Добавить элемент в очередь\n"

<< "3 = Удалить передний элемент из очереди\n"

<< "4 = Ввести данные из файла в очередь\n"

<< "5 = Реализация \"Псевдозминной\" с очередью\n"

<< "6 = Реализация \"Псевдозминной\" с человеком\n"

<< "7 = Показать передний элемент очереди\n"

<< "8 = Рассортировать по приоритету\n"

<< "9 = Перейти в управление к новой очереди\n"

<< "10 = Слияние очередей\n"

<< "-1 = Завершить и выйти\n\n"

<< "Введите номер функции ";

cin >> nfun;

system("cls");

switch (nfun)

{

case 0:

cout << "Введите путь к файлу куда записать очередь\n";

cin.ignore();

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

getline(cin, filename);

qarr[nq].file(filename);

cout << "Успешно записано\n";

system("pause");

break;

case 1:

qarr[nq].show();

system("pause");

break;

case 2:

cout << "Введите имя и приоритет, которое добавим в очередь\nИмя: ";

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cin >> name;

cin.clear();

cin.ignore();

cout << "Приритет: "; cin >> priority;

p.name = name;

p.priority = priority;

qarr[nq].push(p);

system("pause");

break;

case 3:

cout << endl;

qarr[nq].pop();

system("pause");

break;

case 4:

cout << "Введите имя файла ";

cin.clear();

cin.ignore();

getline(cin, filename);

qarr[nq].getfromfile(filename);

break;

case 5:

cout << "Введите границы очереди для изменения\n"

<< "Левая "; cin >> l; cin.clear();

cout << "Правая "; cin >> r; cin.clear();

cout << "Введите номер очереди которую вставить в данную ";

cin >> i; cin.clear();

cout << "Введите границы из добавляемой очереди\n"

<< "Левая "; cin >> l1; cin.clear();

cout << "Правая "; cin >> r1; cin.clear();

qarr[nq](l, r) = qarr[i](l1, r1);

break;

case 6:

cout << "Введите границы очереди для изменения\n"

<< "Левая "; cin >> l;

cin.clear();

cout << "Правая "; cin >> r;

cin.clear();

cout << "Введите данные о человеке\n"

<< "Имя ";

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cin >> name;

p.name = name;

cin.clear();

cout << "Приоритет "; cin >> priority;

p.priority = priority;

qarr[nq](l, r) = p;

break;

case 7:

cout << "Имя: " << qarr[nq].peek().name << "\t\tПриоритет: " << qarr[nq].peek().priority << endl;

system("pause");

break;

case 8:

qarr[nq].setpriority();

cout << "Отсортированно\n";

break;

case 9:

cout << "Введите номер нужной очереди ";

cin >> nq;

cout << "Управление передано\n";

system("pause");

break;

case 10:

cout << "Введите номер очереди для сливания ";

cin >> i;

qarr[nq] + qarr[i];

cout << "Слияние произошло\n";

system("pause");

break;

default:

break;

}

cin.clear();

filename.clear();

system("cls");

}

SetConsoleCP(866);

SetConsoleOutputCP(866);

return 0;

}

Опис інтерфейсу програми(діалог з користувачем)

Діалог з користувачем йде у вигляді меню, у якому ми можемо зробити декілька основних маніпуляцій з нашим класом, а саме чергою, реалізувати роботу псевдозмінної та записати дані на диск(так само і зчитати).

Аналіз завдання

Завдяки такому завданню, я зрозумів як користуватися основними у програмуванні методами розв’язання задач наприклад роботу користувача з його файловою системою і моїми об’єктами(черг) та можливість використання нових перевантажених операторів у мові С++.