ГУО “БГУИР”

Факультет компьютерного проектирования

Кафедра Инженерной психологии и эргономики

Отчёт по

Лабораторной работе №3

СРЕДА ПРОГРАММИРОВАНИЯ VISUAL C++. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Подготовил:

Студент гр.110101

Ладутько Я.Д.

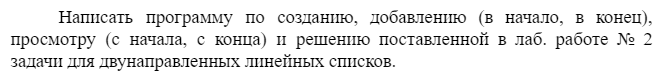
Проверила:

Семижон Е.А.

Минск 2022

***Цель работы:*** изучить возможности работы со списками, организованными в виде очереди.

Вариант 16.



Код:

#include <iostream>

#include <random>

#include <ctime>

using namespace std;

template <typename T>

struct queueData

{

T info = NULL;

queueData<T>\* next\_ptr = nullptr;

queueData<T>\* prew\_ptr = nullptr;

};

template <typename T>

class Queue

{

public:

queueData<T>\* \_top = nullptr;

queueData<T>\* \_end = nullptr;

~Queue()

{

while(\_top != nullptr)

pop();

}

T top()

{

T a = \_top->info;

return a;

}

void push(T x)

{

if(\_top == nullptr)

{

\_top = new queueData<T>();

\_top->info = x;

\_end = \_top;

}

else

{

queueData<T> \* new\_elem = new queueData<T>();

new\_elem->info = x;

new\_elem->prew\_ptr = \_end;

\_end->next\_ptr = new\_elem;

\_end = new\_elem;

}

}

void push\_top(T x)

{

queueData<T>\* prew\_top = \_top;

if (\_top == nullptr)

{

\_top = new queueData<T>();

\_top->info = x;

}

else

{

\_top = new queueData<T>();

\_top->next\_ptr = prew\_top;

\_top->info = x;

prew\_top->prew\_ptr = \_top;

}

}

void pop()

{

if(\_top!=nullptr)

{

queueData<T>\* prew\_top = \_top;

\_top = \_top->next\_ptr;

\_top->prew\_ptr = nullptr;

delete prew\_top;

}

}

void outQueue()

{

queueData<T>\* n\_top = \_top;

while(n\_top!=nullptr)

{

cout << n\_top->info << endl;

n\_top = n\_top->next\_ptr;

}

}

void outQueueBack()

{

queueData<T>\* n\_end = \_end;

while (n\_end != nullptr)

{

cout << n\_end->info << endl;

n\_end = n\_end->prew\_ptr;

}

}

};

template <typename T>

void input(T& a)

{

while (true)

{

cin >> a;

if (cin.fail() || (cin.peek() != '\n' && cin.peek() != ' ' && cin.peek() != '\t'))

{

cin.clear();

cin.ignore(10000, '\n');

cout << "Неверный ввод" << endl;

}

else

{

return;

}

}

}

void input(int& x, int a, int b)

{

while (true)

{

input(x);

if (x < a || x > b)

{

cout <<

"Неверный ввод\n"

"Введите число на промежутке от " << a << " до " << b << endl;

}

else

return;

}

}

void input(bool& b)

{

char c;

while (true)

{

input(c);

if (c != 'y' && c != 'n')

{

cout <<

"Неверный ввод\n"

"Введите y или n\n";

}

else

{

b = (c == 'y' ? true : false);

return;

}

}

}

int main()

{

srand(time(nullptr));

setlocale(LC\_ALL, "RU");

Queue<int> queue;

int n, t;

bool choice;

cout << "Введите количество элементов" << endl;

input(n, 1, 100000);

cout << "Желаете вводить элементы массива вручную? y/n\n";

input(choice);

if (choice)

{

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

{

input(t);

queue.push(t);

}

}

else

{

int random\_min, random\_max;

cout << "Введите минимальное случайное значение\n";

input(random\_min);

cout << "Введите максимальное случайное значение\n";

input(random\_max);

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

{

queue.push(rand() % (random\_max - random\_min + 1) + random\_min);

}

cout << "Список:\n";

queue.outQueue();

}

int sum = 0, avrg;

queueData<int>\* n\_top = queue.\_top;

while (n\_top != nullptr)

{

sum += n\_top->info;

n\_top = n\_top->next\_ptr;

}

avrg = sum / n;

cout << "Среднее: " << avrg << endl;

n\_top = queue.\_top;

while (n\_top != nullptr)

{

if (n\_top->info % 2 == 0)

n\_top->info = avrg;

n\_top = n\_top->next\_ptr;

}

cout << "Конечный список:\n";

queue.outQueue();

}

Вывод: изучил возможности работы со списками, организованными в виде очереди.

