**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ**

**Кафедра інформаційної безпеки**

Комп’ютерний практикум №8

З дисципліни Програмування:

**«Використання зв’язних списків»**

Варіант 11

|  |  |
| --- | --- |
| Перевірив(-ла):  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Виконав:  Студент 1 курсу ФТІ  Групи ФБ-91  Красний Павло Володимирович  № Залікової книжки: ФБ-9112 |

Київ 2019

**Мета роботи**:

Отримати навички реалізації абстрактних типів даних, динамічних структур даних та засвоїти основні принципи роздільної компіляції.

**Завдання:**

Використовувати (лінійні) односпрямовані списки без заголовної ланки (мал. а) або з заголовною ланкою (мал. б) при наступному їхньому описі Параметр L позначає список, а параметри Е, Е1 — дані типу ТЕ, до яких можна застосовувати операції присвоювання і перевірки на рівність. {ТЕ=int} Непорожня послідовність натуральних чисел вводиться з екрану, за якої випливає 0. Визначити функції, що:

а) виводить числа в зворотному порядку ;

б) виводить порядкові номери тих чисел послідовності, що мають

найбільшу величину;

в) сортує числа в порядку їхнього неспадання;

г) записує список у файл

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип** | **Призначення** |
| n | int | Число, яке ми додаємо до списку |
| m | int | Номер функції в інтеракт. меню |
| size | int | Кількість елементів списку |
| head | Node\* | Вказівник на перший елемент(вузол) списку |
| pNext | Node\* | Вказівник на адресу наступного елемента в списку |
| current | Node\* | Вказівник на поточний елемент |
| towrite | ofstream | Змінна для запису в файл |
| flag | bool | Змінна для перевірки відсортованості |
| max | int | Змінна для знаходження найбільшого числа |
| list | struct | Однозв’язний список |
| list | - | Конструктор |
| ~list | - | Деструктор |
| Node | struct | Елемент списку(вузол) |
| push\_back() | void | Функція додавання введеного числа в кінець списку |
| GetSize() | int | Функція отримання розміру списку |
| operator[] | int& | Перегрузка оператора [], що дає змогу працювати з Nod`ами по індексу |
| clear() | void | Функція очищення пам’яті |
| reverse() | void | Функція виводу послідовності в зворотньому порядку |
| sort() | void | Функція сортування послідовності в порядку неспадання і її виводу |
| MaxIndex() | void | Функція знаходження найбільших елементів та вивід їх індексів |
| toWrite() | void | Функція запису послідовності у файл |
| toRead() | void | Функція отримання послідовності з файлу |

# laba8.cpp

#include <iostream>

#include"list.h"

#include"funcs.h"

using namespace std;

int main(){

system("chcp 1251");

cout << "\nПривіт! Обери функцію:\n1 - Ввести список з клавіатури\n2 - Отримати список з файлу\n3 - Вивести всі числа в зворотньому порядку.\n4 - Вивести порядкові номери елементів, що мають найбільшу величину.\n5 - Сортує числа в порядку їхнього неспадання.\n6 - Записати числа у файл.\n0 - Вихід.\n№: ";

int M;

cin >> M;

list list;

while (M){

if (M == 1){

list.clear();

cout << "Введіть послідовність натуральних чисел. 0 - кінець вводу:\n ";

int n;

cin >> n;

while (n){

list.push\_back(n);

cin >> n;

}

}

else if (M == 2){

list.clear();

toRead(list);

}

else if (M == 3){

reverse(list);

}

else if (M == 4){

MaxIndex(list);

}

else if (M == 5){

sort(list);

}

else if (M == 6){

toWrite(list);

}

else{

cout << "Оберіть правильний номер: ";

cin >> M;

continue;

}

cout << "\nСпробуйте ще:\n";

cin >> M;

}return 0;}

# list.h

#pragma once

struct list

{

public:

list();

void push\_back(int data);

int GetSize();

int& operator[](const int index);

void pop\_front();

void clear();

~list();

private:

struct Node;

Node\* head;

int size;

};

struct list:: Node

{

Node\* pNext;

int data;

Node(int data = 0)

{

this->data = data;

this->pNext = nullptr;

}

};

# funcs.h

#pragma once

#include<iostream>

#include"list.h"

#include <fstream>

using namespace std;

void toRead(list& list);

void reverse(list& list);

void sort(list& list);

void MaxIndex(list& list);

void toWrite(list& list);

# list.cpp

#include "list.h"

list::list()

{

size = 0;

head = nullptr;

}

list::~list()

{

clear();

}

int list::GetSize()

{

return size;

}

void list::push\_back(int data)

{

if (head == nullptr)

{

head = new Node(data);

}

else

{

Node\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Node(data);

}

size++;

}

int& list::operator[](const int index)

{

int counter = 0;

Node\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

return current->data;

}

current = current->pNext;

counter++;

}

}}

void list::pop\_front()

{

Node\* temp = head;

head = head->pNext;

delete temp;

size--;

}

void list::clear()

{

while (size)

{

pop\_front();

}

}

# toRead.cpp

#include"funcs.h"

void toRead(list& list)

{

cout << "Введіть назву файлу, з якого буде прочитано список: ";

try

{

string name;

cin >> name;

ifstream toread;

toread.open(name);

if (!toread.is\_open())

{

cout << "Файл не вдалося відкрити";

throw 1;

}

else {

toread.seekg(0, ios\_base::end);

if (!toread.tellg())

{

cout<< "Файл порожній";

throw "gg";

}

toread.seekg(0, ios\_base::beg);

cout << "Список успішно зчитаний:\n";

while (!toread.eof())

{

int temp;

toread >> temp;

list.push\_back(temp);

cout << temp << ' ';

}

}

}

catch (...)

{

}

}

# MaxIndex.cpp

#include"funcs.h"

void MaxIndex(list& list)

{

int max = list[0];

for (int i = 0; i < list.GetSize(); i++)

{

if (list[i] > max) { max = list[i]; }

}

for (int i = 0; i < list.GetSize(); i++)

{

if (list[i] == max)

{

cout << "Елемент найбільшої величини має номер " << i + 1 << endl;

}

}

}

# reverse.cpp

#include"funcs.h"

void reverse(list& list)

{

for (int i = list.GetSize() - 1; i >= 0; i--)

{

cout << list[i] << ' ';

}

}

# sort.cpp

#include"funcs.h"

void sort(list& list)

{

for (int i = 0; i < list.GetSize() - 1; i++)

{

bool flag = false;

for (int j = 0; j < list.GetSize() - 1 - i; j++)

{

if (list[j] > list[j + 1])

{

list[j] = list[j] + list[j + 1];

list[j + 1] = list[j] - list[j + 1];

list[j] = list[j] - list[j + 1];

flag = true;

}

}

if (!flag)

{

break;

}

}

for (int i = 0; i < list.GetSize(); i++)

{

cout << list[i] << ' ';

}

}

# toWrite.cpp

#include"funcs.h"

void toWrite(list& list)

{

ofstream towrite;

towrite.open("file.txt");

if (!towrite.is\_open())

{

cout << "Не вдалося відкрити файл" << endl;

}

else

{

for (int i = 0; i < list.GetSize(); i++)

{

towrite << list[i]<<' ';

}

cout << "Операція прошла успішно";

towrite.close();

}