**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 |  | Чесноков М. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение структур в c++, массивов структур, создание прототипа баз данный в c++ при помощи структур, добавление функционала.

**Основные теоретические положения.**

В языке c++, структура — композитный тип данных, инкапсулирующий без сокрытия набор значений различных типов. Порядок размещения значений в памяти задаётся при определении типа и сохраняется на протяжении времени жизни объектов, что даёт возможность косвенного доступа. Пример объявления структуры: struct str\_name

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

**Постановка задачи**

Необходимо создать программу, позволяющую:

1.   Создание новой записи о студенте.

2.   Внесение изменений в уже имеющуюся запись.

3.   Вывод всех данных о студентах.

4.   Вывод информации обо всех студентах группы *N*. *N* – инициализируется пользователем.

5.   Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.

6.   Вывод количества студентов мужского и женского пола.

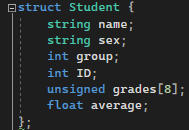
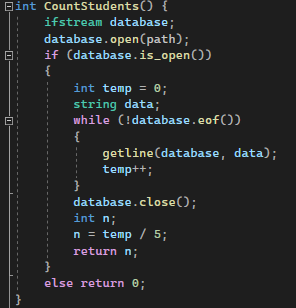
7.   Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;

8.   Вывод данных о студентах, имеющих номер в списке – *k*.

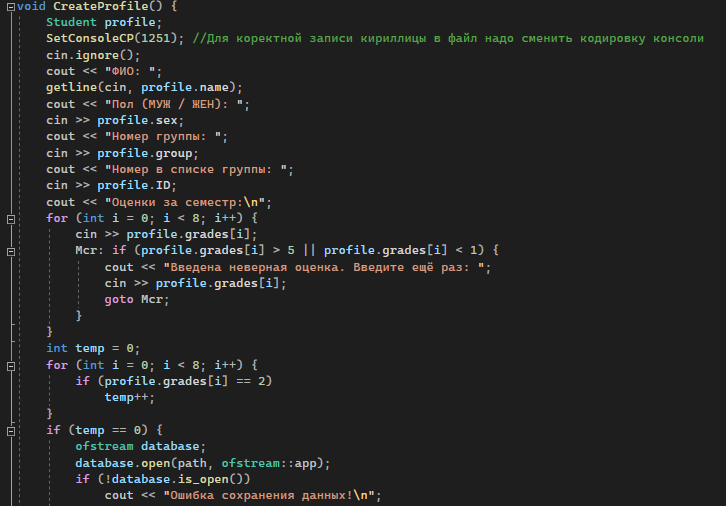
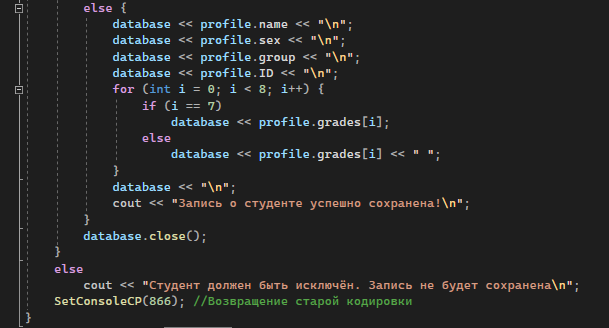
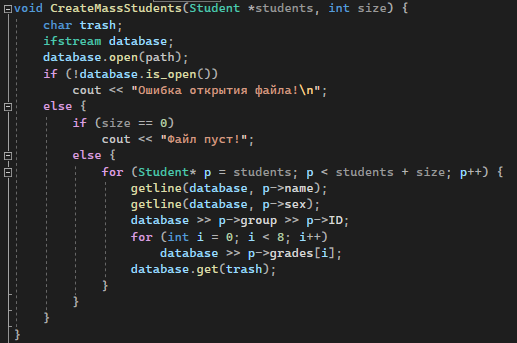
**Выполнение работы.**

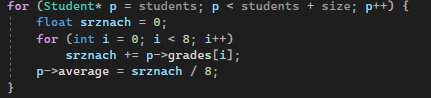
Код программы представлен в приложении А.

**Блок описания кода и использованных алгоритмов:**

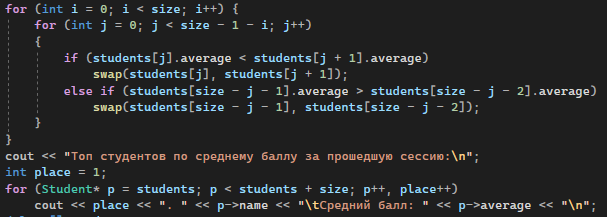
Создаём динамический массив структур, размер которого узнаём, считывая количество строк в изначальном файле Students.txt.

1. Создаётся элемент структуры, данные в который записывает пользователь, и сохраняется в файл.

Перед выполнением любого другого пункта программы создаётся динамический массив, который удаляется после выполнения пункта.

1. Пользователь вводит имя студента, данные которого хочет изменить, а затем меняет их.
2. Циклом for по очереди выводим всех студентов в списке.
3. Пользователь вводит номер группы и перебором выводятся студенты с соответствующим номером группы.
4. Сначала создаём функцию для высчитывания среднего значения оценок обучающегося.

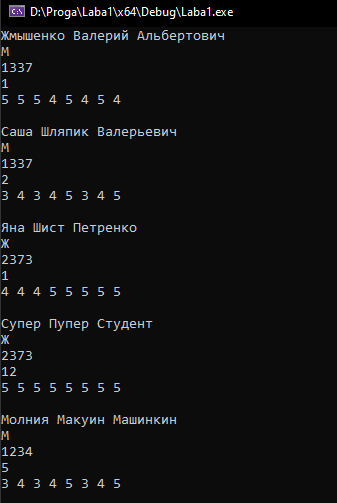
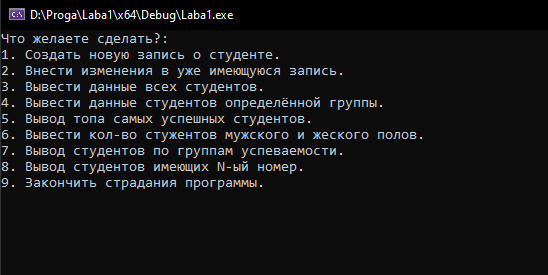
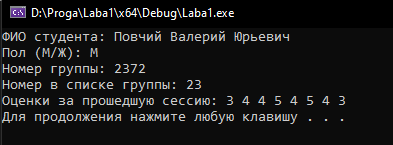
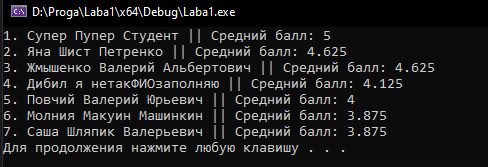
Далее сортируем массив алгоритмом Bubble sort.



1. Перебором проходимся по всему массиву и увеличиваем счётчики обеих полов.
2. Делаем проверку оценок студентов. Если находим соответствие, то выводим в консоль.
3. Пользователь вводит номер и перебором выводятся студенты с соответствующим номеров в группе.

**Выводы.**

Мы научились использовать структуры c++, написали прототип баз данный c++, научились использовать функционал структур.

**Блок скриншотов работы программы**

Приложение А

рабочий код

#include <iostream>

#include <string>

#include <chrono>

using namespace std;

using namespace chrono;

struct List {

int data;

List\* prev;

List\* next;

};

struct Timer {

time\_point<steady\_clock, duration<\_\_int64, ratio<1, 1000000000>>> start, end;

duration<\_\_int64, ratio<1, 1000000000>> duration;

} timer;

List\* CreateRandList(int& lenght) {

List\* curr = NULL, \* prev = NULL;

timer.start = steady\_clock::now();

if (lenght == 0) {

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

return curr;

}

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

curr = new List;

curr->data = rand() % 100;

curr->prev = prev;

if (prev)

prev->next = curr;

prev = curr;

}

curr->next = NULL;

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

return curr;

}

List\* CreateInputList(int& lenght) {

List\* curr = NULL, \* prev = NULL;

string inp;

int data;

timer.start = steady\_clock::now();

while (true) {

cout << "Введите элемент списка (000 - закончить ввод): ";

cin >> inp;

if (inp == "000")

break;

data = stoi(inp);

lenght++;

curr = new List;

curr->data = data;

curr->prev = prev;

if (prev)

prev->next = curr;

prev = curr;

}

if (lenght == 0) {

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

return curr;

}

curr->next = NULL;

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

return curr;

}

List\* DelAllList(List\* curr) {

List\* del = curr;

while (curr) {

curr = curr->next;

delete del;

del = curr;

}

return curr;

}

void ShowList(List\* curr, int& lenght) {

while (curr) {

cout << curr->data << " ";

curr = curr->next;

}

cout << "\nДлина списка: " << lenght << "\n";

}

List\* Add(List\* curr, int& lenght) {

List\* prev = NULL, \* next = NULL;

int index, data;

cout << "Введите индекс элемента, на место которого хотите вставить новый элемент: ";

cin >> index;

if (index > lenght || index < 0) {

cout << "Данный индекс выходит за пределы списка. Вы будете возвращены в меню.";

return curr;

}

cout << "Введите значение нового элемента списка: ";

cin >> data;

timer.start = steady\_clock::now();

if (lenght == 0) {

curr = new List;

curr->data = data;

curr->prev = NULL;

curr->next = NULL;

lenght++;

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

ShowList(curr, lenght);

return curr;

}

for (int i = 0; i < lenght - 1; i++) {

if (i == index) {

prev = curr->prev;

next = curr;

curr = new List;

curr->data = data;

curr->prev = prev;

curr->next = next;

next->prev = curr;

if (i != 0)

prev->next = curr;

lenght++;

break;

}

curr = curr->next;

}

if (index == lenght) {

prev = curr;

curr = new List;

curr->data = data;

prev->next = curr;

curr->prev = prev;

curr->next = NULL;

lenght++;

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

cout << "Список: ";

ShowList(curr, lenght);

return curr;

}

List\* DelIndex(List\* curr, int& lenght) {

List\* prev = NULL, \* next = NULL;

int index;

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> index;

if (index >= lenght || index < 0) {

cout << "Данный индекс выходит за пределы списка. Вы будете возвращены в меню.\n";

return curr;

}

timer.start = steady\_clock::now();

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

if (i == index) {

prev = curr->prev;

next = curr->next;

delete curr;

curr = NULL;

if (i != 0) {

prev->next = next;

curr = prev;

}

if (i != lenght - 1) {

next->prev = prev;

curr = next;

}

lenght--;

break;

}

curr = curr->next;

}

if (lenght == 0) {

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

cout << "Список пуст!\n";

return curr;

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

cout << "Список: ";

ShowList(curr, lenght);

return curr;

}

List\* DelData(List\* curr, int& lenght) {

List\* prev = NULL, \* next = NULL;

int data;

cout << "Введите значение, все элементы с которым будут удалены: ";

cin >> data;

bool check = 1;

timer.start = steady\_clock::now();

while (check) {

check = 0;

while (curr->prev)

curr = curr->prev;

for (int i = 0; i < lenght - 1; i++) {

if (curr->data == data) {

prev = curr->prev;

next = curr->next;

delete curr;

check = 1;

if (i != 0)

prev->next = next;

next->prev = prev;

curr = next;

lenght--;

break;

}

curr = curr->next;

}

}

if (curr->data == data) {

prev = curr->prev;

if (lenght != 1)

prev->next = NULL;

delete curr;

curr = prev;

lenght--;

}

if (lenght == 0) {

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

cout << "Список пуст!\n";

return curr;

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

cout << "Список: ";

ShowList(curr, lenght);

return curr;

}

List\* Swap(List\* curr, int& lenght) {

int index1, index2;

List\* swap1 = NULL, \* swap2 = NULL, \* sp1 = NULL, \* sp2 = NULL, \* sn1 = NULL, \* sn2 = NULL;

cout << "Введите индексы элементов списка, которые желаете обменять.\n1 индекс: ";

cin >> index1;

cout << "2 индекс: ";

cin >> index2;

if (index1 >= lenght || index1 < 0 || index2 >= lenght || index2 < 0) {

cout << "Один из индексов выходит за пределы списка. Вы будете возвращены в меню.\n";

return curr;

}

if (index1 == index2) {

cout << "Индексы не могут совпадать. Вы будете возвращены в меню.\n";

return curr;

}

if (index1 > index2)

swap(index1, index2);

timer.start = steady\_clock::now();

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

if (i == index1) {

swap1 = curr;

sp1 = swap1->prev;

sn1 = swap1->next;

}

if (i == index2) {

swap2 = curr;

sp2 = swap2->prev;

sn2 = swap2->next;

}

if (swap1 && swap2)

break;

curr = curr->next;

}

if (index2 - index1 == 1) {

if (index1 != 0)

sp1->next = swap2;

if (index2 != lenght - 1)

sn2->prev = swap1;

swap1->next = sn2;

swap2->next = swap1;

swap1->prev = swap2;

swap2->prev = sp1;

}

else {

swap(swap1->next, swap2->next);

swap(swap1->prev, swap2->prev);

if (!sp1 && !sn2) {

sn1->prev = swap2;

sp2->next = swap1;

}

else if (!sp1) {

swap(sn1->prev, sn2->prev);

sp2->next = swap1;

}

else if (!sn2) {

swap(sp1->next, sp2->next);

sn1->prev = swap2;

}

else {

swap(sn1->prev, sn2->prev);

swap(sp1->next, sp2->next);

}

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

while (curr->prev)

curr = curr->prev;

cout << "Список: ";

ShowList(curr, lenght);

return curr;

}

void GetIndex(List\* curr, int& lenght) {

int index;

cout << "Введите индекс элемента, который хотите получить: ";

cin >> index;

if (index >= lenght || index < 0) {

cout << "Данный индекс выходит за пределы списка. Вы будете возвращены в меню.\n";

return;

}

timer.start = steady\_clock::now();

int count = 0;

while (count != index) {

curr = curr->next;

count++;

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

cout << "Значение: " << curr->data << "\tИндекс: " << count << "\tАдрес в памяти: " << curr << "\n";

}

void GetData(List\* curr, int& lenght) {

int data;

cout << "Введите значение, все элементы с которым вы хотите получить: ";

cin >> data;

timer.start = steady\_clock::now();

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

if (curr->data == data) {

cout << "Значение: " << curr->data << "\tИндекс: " << i << "\tАдрес в памяти: " << curr << "\n";

}

curr = curr->next;

}

timer.end = steady\_clock::now();

timer.duration = timer.end - timer.start;

}

int main() {

srand(time(NULL));

setlocale(0, "");

List\* list = NULL;

int lenght = 0;

char menu;

while (true) {

cout << "\tМеню\n"

<< "1 - Создать новый список (существующий список будет удалён).\n"

<< "2 - Вывести список.\n"

<< "3 - Вставить элемент в список.\n"

<< "4 - Удалить элемент списка.\n"

<< "5 - Поменять 2 элемента списка местами.\n"

<< "6 - Получить элемент списка.\n"

<< "0 - Выйти из программы\n"

<< "Выберите пункт меню: ";

cin >> menu;

switch (menu) {

case '1':

system("cls");

if (lenght != 0) {

list = DelAllList(list);

lenght = 0;

}

char create;

cout << "1 - Создать двусвязный список со случайными числами от 0 до 99.\n"

<< "2 - Создать список и ввести элементы вручную (длина списка определится автоматически).\n"

<< "Ваш выбор: ";

cin >> create;

if (create == '1') {

cout << "Введите длину списка: ";

cin >> lenght;

list = CreateRandList(lenght);

cout << "Создан список: ";

ShowList(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else if (create == '2') {

list = CreateInputList(lenght);

cout << "Создан список: ";

ShowList(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else

cout << "Некорректный выбор. Вы будете возвращены в меню.\n";

system("pause");

system("cls");

break;

case '2':

system("cls");

if (lenght == 0) {

cout << "Список пуст!\n";

system("pause");

system("cls");

break;

}

cout << "Список: ";

ShowList(list, lenght);

system("pause");

system("cls");

break;

case '3':

system("cls");

list = Add(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

system("pause");

system("cls");

break;

case '4':

system("cls");

if (lenght == 0) {

cout << "Список пуст!\n";

system("pause");

system("cls");

break;

}

cout << "1 - Удалить элемент по инедксу.\n"

<< "2 - Удалить элементы по значению (удалятся все элементы с данным значением).\n"

<< "Ваш выбор: ";

cin >> create;

if (create == '1') {

list = DelIndex(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else if (create == '2') {

list = DelData(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else

cout << "Некорректный выбор. Вы будете возвращены в меню.\n";

system("pause");

system("cls");

break;

case '5':

system("cls");

if (lenght < 2) {

cout << "Список должен состоять хотя бы из 2 элементов!\n";

system("pause");

system("cls");

break;

}

list = Swap(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

system("pause");

system("cls");

break;

break;

case '6':

system("cls");

if (lenght == 0) {

cout << "Список пуст!\n";

system("pause");

system("cls");

break;

}

cout << "1 - Получить элемент по инедксу.\n"

<< "2 - Получить элементы по значению (выведятся все элементы с данным значением).\n"

<< "Ваш выбор: ";

cin >> create;

if (create == '1') {

GetIndex(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else if (create == '2') {

GetData(list, lenght);

cout << "Время выполнения: " << timer.duration.count() << " наносек.\n";

}

else

cout << "Некорректный выбор. Вы будете возвращены в меню.\n";

system("pause");

system("cls");

break;

case '0':

return 0;

default:

cout << "Пункт меню выбран неправильно!\n";

system("pause");

system("cls");

cin.ignore();

}

}

}