|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 4** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Структуризация многоэлементных структур средствами struct.** | |
|  | |
| Выполнил студент группы ИКБО-72-23 | Шатохин Б.А. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

1. Цель работы

Получение навыков по реализации многоэлементных структур данных задачи. Получение навыков по реализации многоэлементных данных средствами структуры данных – таблица.

1. Постановка задачи

Разработать набор операций для управления таблицей, созданной на основе динамического массива. Структура записи элемента таблицы определена вариантом индивидуального задания.

1. Создать проект, добавить заголовочный файл. Включить в заголовочный файл:
2. разработанный тип данных, определяющий структуру элемента таблицы (записи).
3. определенная на базе типа struct структура хранения данных по таблице, содержащая: размер таблицы – n, массив из n элементов типа записи варианта.
4. реализованная функция вывода таблицы на экран.
5. операции варианта задания, оформленные как функции с параметрами.
6. Для заполнения отдельной записи с клавиатуры разработать функцию, которая принимает в качестве параметра пустую запись, а возвращает заполненную.
7. Наполнение таблицы данными должна выполнять функция вставки или добавления записи, включенная в список операций варианта. Эта функция должна приять через параметр заполненную запись, выполнить действие по вставке или добавления, согласно алгоритму варианта этой операции.

Разработать программу, демонстрирующую работу всех функций с массивом или записей. Программа должна позволять пользователю непрерывно выполнять операции над таблицей в произвольном порядке.

Разработать набор тестовых данных по наполнению таблицы. Включить в меню программы возможность автоматического ввода разработанных тестовых данных.

Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Индивидуальное задание:

Расчет активности на основе выполнения индивидуальных заданий.

Структура записи по выполненным заданиям: код студента (номер зачетной книжки, номер задания, задача 1, задача 2, задача 3.

Балл за задачу: задача 1 – 10 баллов; задача 2 -20 баллов; задача 3 -60 баллов.

Операции

1) Заполнение данных по заданному заданию заданного студента с клавиатуры.

2) Вставить запись по студенту в начало таблицы.

3) Удалить записи по заданному студенту.

4) Сформировать таблицу с результатами выполнения заданий заданным студентом.

1. Решение

В языке Си, 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. В языке C++ существует понятие класса. По своей сути это улучшенный композитный тип данных. Однако для упрощения решения мы будем использовать struct.

Так же для решения будем использовать модуль format, который появился в языке С++ со стандарта С++ 20. Функция std::format(), которой мы будем пользоваться, форматирует строку под удобный для нас вид, тем самым стандартизируя запись.

Для выполнения индивидуального задания была разработана структура данных student, которая содержит поля: student\_id (идентификационный номер студента, который служит ключом), task\_number (номер задания), problem\_1 (задача 1 стоимость 10 баллов), problem\_2 (задача 2 стоимостью 20 баллов), problem\_3 (задача 3 стоимостью 30 баллов) . Так же был разработан параметризированный конструктор, который принимает параметр id. Для ввода полей был разработан метод DataInput. Для удобного вывода данных был разработан метод sCount, который считает общее количество баллов. Так же для вывода данных об объекте был разработан метод print, который использует функцию std::format().

struct student {

int student\_id; //Номер зачётной книжки

int task\_number; //номер задания

//Поля отвечающие за выполненные задачи (по-умол. false)

bool problem\_1 = false; //Задача 1

bool problem\_2 = false; //Задача 2

bool problem\_3 = false; //Задача 3

student(int id) { //Параметризированный конструктор с пар-ом id

student\_id = id;

}

void DataInput() { //Ввод значений для полей структуры

std::cout << "Введите номер задания: ";

std::cin >> task\_number;

std::cout << "Введите решена ли задача\n";

std::cout << "1. ";

std::cin >> problem\_1;

std::cout << "2. ";

std::cin >> problem\_2;

std::cout << "3. ";

std::cin >> problem\_3;

}

int sCount() { //Подсчёт баллов за задание

int total\_sum = 0;

if (problem\_1 == true) total\_sum += 10;

if (problem\_2 == true) total\_sum += 20;

if (problem\_3 == true) total\_sum += 30;

return total\_sum;

}

void print() { //Вывод по шаблону

std::cout << std::format("id студента: {:<5} | задание: {:<5} | задача 1: {:<5} | задача 2: {:<5} | задача 3: {:<5} | кол-во баллов: {:<5}\n",

student\_id, task\_number, problem\_1, problem\_2, problem\_3, sCount());

}

};

Для создания результирующей таблицы была разработана структура данных resulting\_table. В структуре имеются: конструктор по умолчанию, поле rating\_table, метод добавления элементов в таблицу addToTable, метод удаления студента из таблицы deleteFromTable, метод вывода таблицы printTable.

Метод addToTable принимает объект структуры student и добавляет его в начало вектора со всеми студентами, с помощью метода класса vector insert().

Метод deleteFromTable, аналогично методу выше принимает объект структуры student и использует метод erase().

Метод printTable ничего не принимает, однако использует поле rating\_table и метод print структуры student

struct resulting\_table {

std::vector<student> rating\_table; //Создание вектора со студентами

resulting\_table() = default;

void addToTable(student human) { //Метод добавления студента в начало вектора

rating\_table.insert(rating\_table.begin(), human);

}

void deleteFromTable(student human) { //Метод удаления студента из вектора

int n = -1; //Инициализация индекса нужного студента

for (int i = 0; i < rating\_table.size(); i++) { //Алгоритм поиска студента

if (rating\_table[i].student\_id == human.student\_id) {

n = i;

break;

}

}

if (n == -1) { //Проверка на налиичие студента

std::cout << "Неверный id\n";

return;

}

rating\_table.erase(rating\_table.begin()+n); //Удаление студента по индексу в векторе

}

void printTable() { //Метод вывода таблицы

for (int i = 0; i < rating\_table.size(); i++) {

rating\_table[i].print();

}

}

};

Была реализована функция makeStudent, которая создаёт объект структуры student.

student makeStudent(int student\_id) { //Функция создающая объект структуры student

student temp(student\_id);

return temp;

}

1. Тестирование

Для удобного демонстрирования был разработан демо-режим, который можно выбрать в начальном меню.

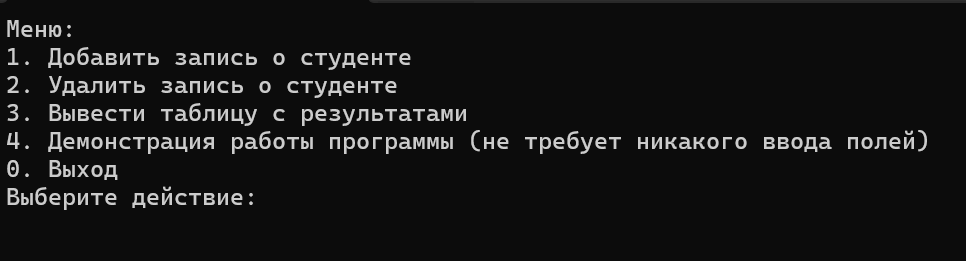


Рисунок 1. Меню программы

Выбираем режим работы “4”

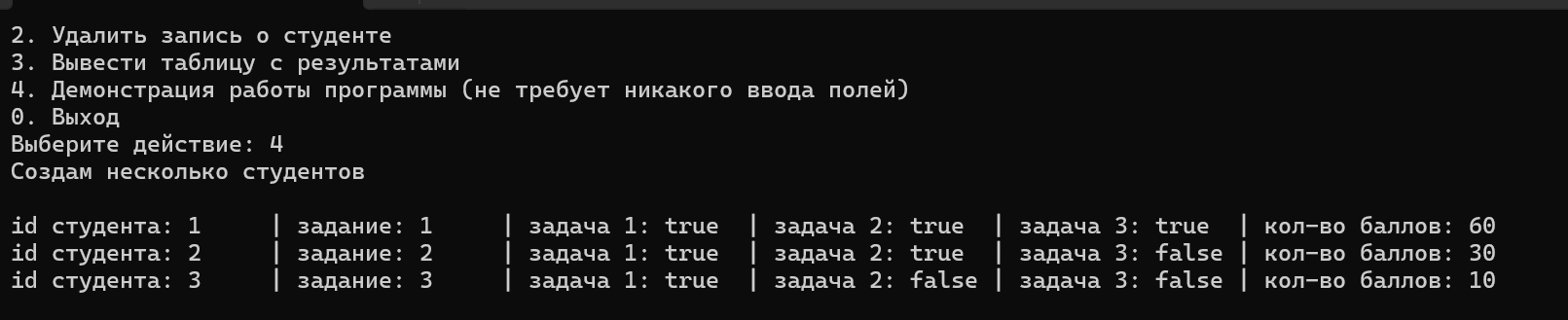


Рисунок 2. Демонстрация функции заполнения программы

Создадим и добавим в начало таблицы нового студента под номером 4.

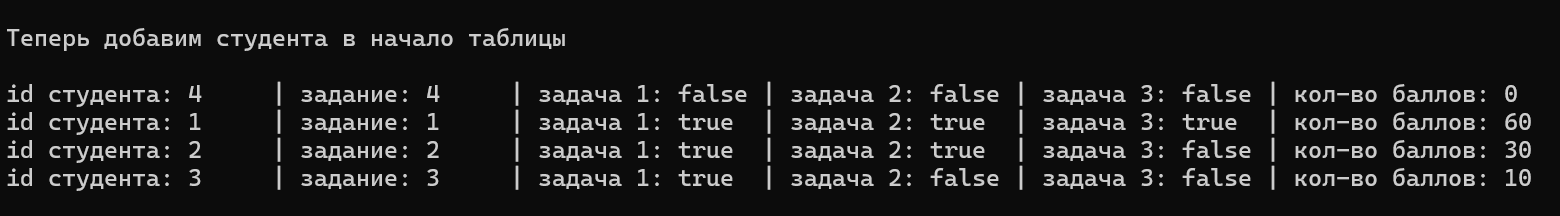


Рисунок 3. Демонстрация возможности добавления элемента в начало таблицы

Теперь необходимо удалить студента, выберем студента с наименьшим количеством баллов.

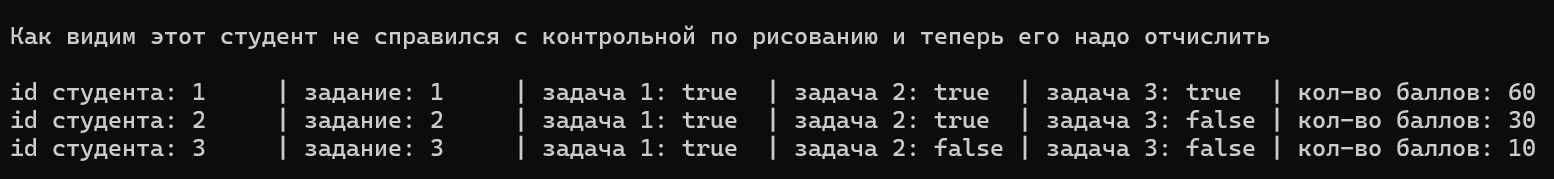


Рисунок 4. Демонстрация возможности удаления элемента таблицы.

1. Вывод

Были получены навыки по работе со структурами данных struct. Так же были изучены методы создания таблицы данных использую функцию std::format и специально созданную структуру данных.

1. Исходный код

int main() {

student student\_1(00001), student\_2(00002), student\_3(00003), student\_4(00004);

resulting\_table test\_table;

setlocale(LC\_ALL, "ru");

std::vector<student> zoo; //Объявление вектора всех студентов

resulting\_table total\_table; //Объявление результирующей таблицы

char menu;

int student\_id;

bool flag = true;

std::cout << "Меню:\n"

<< "1. Добавить запись о студенте\n"

<< "2. Удалить запись о студенте\n"

<< "3. Вывести таблицу с результатами\n"

<< "4. Демонстрация работы программы (не требует никакого ввода полей)\n"

<< "0. Выход\n";

while (true) {

std::cout << "Выберите действие: ";

std::cin >> menu;

switch (menu) {

case '1':

flag = true;

std::cout << "Введите id студента\n";

std::cin >> student\_id;

for (int i = 0; i < zoo.size(); i++) {

if (student\_id == zoo[i].student\_id) { //Проверка уникальности ключа

std::cout << "Такой студент уже есть!!!\n";

flag = false;

break;

}

}

if (flag) { //Создание объекта структуры и добавление в вектор

zoo.push\_back(makeStudent(student\_id));

zoo[zoo.size() - 1].DataInput();

std::cout << "Студент успешно добавлен...\n";

break;

}

else break;

if (flag) { //Создание объекта структуры и добавление в вектор

zoo.push\_back(makeStudent(student\_id));

zoo[zoo.size() - 1].DataInput();

std::cout << "Студент успешно добавлен...\n";

break;

}

else break;

case '2':

flag = true;

std::cout << "Введите id студента\n";

std::cin >> student\_id;

for (int i = 0; i < zoo.size(); i++) {

if (student\_id == zoo[i].student\_id) { //Поиск нужного студента

zoo.erase(zoo.begin() + i); //Удаление студента из вектора всех студентов

std::cout << "Студент успешно удалён!\n";

flag = false;

break;

}

}

if (flag) {

std::cout << "Студент не найден!\n";

break;

}

else break;

case '3':

for (int i = 0; i < zoo.size(); i++) {

total\_table.addToTable(zoo[i]);

}

total\_table.printTable();

break;

case '4':

std::cout << "Создам несколько студентов\n\n";

//Студент не курильщика

student\_3.task\_number = 3;

student\_3.problem\_1 = 1;

student\_3.problem\_2 = 0;

student\_3.problem\_3 = 0;

test\_table.addToTable(student\_3);

//Студент хорошист

student\_2.task\_number = 2;

student\_2.problem\_1 = 1;

student\_2.problem\_2 = 1;

student\_2.problem\_3 = 0;

test\_table.addToTable(student\_2);

//Студент отличник

student\_1.task\_number = 1;

student\_1.problem\_1 = 1;

student\_1.problem\_2 = 1;

student\_1.problem\_3 = 1;

test\_table.addToTable(student\_1);

test\_table.printTable();

std::cout << "\nТеперь добавим студента в начало таблицы\n\n";

//Добавляемы студент

student\_4.task\_number = 4;

student\_4.problem\_1 = 0;

student\_4.problem\_2 = 0;

student\_4.problem\_3 = 0;

test\_table.addToTable(student\_4);

test\_table.printTable();

std::cout << "\nКак видим этот студент не справился с контрольной по рисованию и теперь его надо отчислить\n\n";

test\_table.deleteFromTable(student\_4);

test\_table.printTable();

std::cout << "\n";

case '0':

std::cout << "До свидания!\n";

return 1;

default:

std::cout << "Неверно выбран вариант!\n";

std::cout << "Меню:\n"

<< "1. Добавить запись о студенте\n"

<< "2. Удалить запись о студенте\n"

<< "3. Вывести таблицу с результатами\n"

<< "4. Демонстрация работы программы (не требует никакого ввода полей)\n"

<< "0. Выход\n";

}

}

}

//Добавляемы студент

student\_4.task\_number = 4;

student\_4.problem\_1 = 0;

student\_4.problem\_2 = 0;

student\_4.problem\_3 = 0;

test\_table.addToTable(student\_4);

test\_table.printTable();

std::cout << "\nКак видим этот студент не справился с контрольной по рисованию и теперь его надо отчислить\n\n";

test\_table.deleteFromTable(student\_4);

test\_table.printTable();

std::cout << "\n";

case '0':

std::cout << "До свидания!\n";

return 1;

default:

std::cout << "Неверно выбран вариант!\n";

std::cout << "Меню:\n"

<< "1. Добавить запись о студенте\n"

<< "2. Удалить запись о студенте\n"

<< "3. Вывести таблицу с результатами\n"

<< "4. Демонстрация работы программы (не требует никакого ввода полей)\n"

<< "0. Выход\n";

}

}

}