СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Человек в своей жизни использует большие объемы информации. Множество фактов, необходимых человеку достаточно велико. Но человек использует не весь объем своей памяти, поэтому у большинства людей часто наступает момент, когда человек жалуется на то, что нужная информация “вылетела из головы”. Чтобы избежать этого, люди прибегают к записям. Однако недостаточно только лишь записать информацию – необходимо уметь быстро ее найти в нужное время, т.е. тогда, когда это необходимо. Для хранения и облегчения поиска нужной информации человеком придуманы различные способы. К ним относятся записные книжки, ежедневники, перекидные календари. Такие устройства называются базами данных.

База данных - это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

На сегодняшний день применение баз данных приобрело весьма важное значение для многих организаций, будь то какая-либо компания или учреждение образования. Ради улучшения производительности и качества работы с большим количеством данных используют базы данных.

Основная цель работы – создание консольного приложения, предоставляющего пользователю инструменты для работы с массивом структурированных данных, содержащем в себе информацию об абитуриентах ВУЗа.

1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать программу, осуществляющую обработку массивов структурированных данных «Список абитуриентов ВУЗа». Структура должна иметь не менее пяти полей (элементов) двух или более типов, включая пользовательский тип union и enum. Также должны присутствовать следующие поля: средний балл аттестата (float), фамилия (char[]), год рождения (int). Программа должна быть обязательно реализована в виде нескольких модулей.

Работа содержит описание разработанного программного обеспечения по обработке массива структур, обеспечивающего реализацию следующих запросов к заданному массиву структурированной информации:

1. Ввод информации из текстового файла в массив указателей на записи;
2. Добавление новых элементов в конец массива;
3. Просмотр всех элементов массива;
4. Вывод информации из массива в текстовый файл;
5. Корректировка полей выбранного элемента;
6. Удаление выбранного элемента;
7. Удаление элементов по условию (поле < или > заданного значения);
8. Замена выбранного элемента;
9. Удаление элементов, начиная от выбранного.

Условия и ограничения:

1. Главную процедуру программы с реализацией простейшего меню следует определить в отдельном модуле.
2. Процедуры, реализующие запросы, должны быть размещены в одном или более модулях.
3. Глобальные данные использовать нельзя.
4. На экран выводить элементы в виде таблицы (один элемент – одна строка таблицы).
5. Если после выполнения запроса изменяется хотя бы один элемент, то заканчивать запрос выводом таблицы на экран.
6. Тестами к заданиям служат 2 текстовых файла с правдоподобной информацией.

2 РАЗРАБОТКА АЛГОРИТМОВ

2.1 Алгоритмы общего задания

Алгоритм ввода информации из текстового файла в массив указателей

Исходные данные: текстовый файл, массив записей
Алгоритм:

1. Открытие файла в режиме чтения.
2. Если файл открыт:
    2.1. Цикл, пока не достигнут конец файла:
        2.1.1. Считывание записи из файла построчно.
    2.2. Закрытие файла.
3. Иначе:
    3.1. Цикл:
        3.1.1. Ввод данных для новой записи с клавиатуры.
        3.1.2. Создание новой записи.
        3.1.3. Выход из цикла.
    3.2. Запись данных в файл.

Выходные данные: сформированный массив записей

Алгоритм добавления новых элементов в конец массива

Исходные данные: массив записей
Алгоритм:

1. Цикл:
    1.1. Ввод данных для новой записи с клавиатуры.
    1.2. Создание нового элемента (новой записи).
    1.3. Если создание записей не продолжается, выход из цикла.
2. Вывод данных в файл.

Выходные данные: сформированный массив записей.

Алгоритм просмотра всех элементов списка

Исходные данные: массив записей
Алгоритм:

1. Вывод шапки таблицы.
2. Цикл, пока не достигнут конец массива:
    2.1. Построчный вывод записей из массива в консоль.

Выходные данные: нет.

Алгоритм вывода информации в текстовый файл

Исходные данные: текстовый файл, массив записей

Алгоритм:

1. Открытие файла в режиме записи.
2. Цикл, пока не достигнут конец файла:
    2.1. Запись шапки в файл.
    2.2. Построчная запись данных из массива указателей в файл.
3. Закрытие файла.

Выходные данные: данные текстового файла.

Алгоритм корректировки полей выбранного элемента

Исходные данные: массив записей

Алгоритм:

1. Ввод фамилии абитуриента, чью запись необходимо скорректировать.
2. Цикл, пока не будет достигнут последний элемент массива:
    2.1. Если запись с данной фамилией найдена:
        2.1.1.1. Выбор поля, которое нужно скорректировать.
        2.1.1.2. Ввод новых данных в выбранное поле с клавиатуры.

Выходные данные: скорректированный массив записей.

Алгоритм удаления выбранного элемента

Исходные данные: массив записей

Алгоритм:

1. Ввод фамилии абитуриента, чью запись необходимо удалить.
2. Цикл, пока не будет достигнут последний элемент массива:
    2.1. Если запись с данной фамилией найдена:
        2.1.1. Уменьшение размера массива на одну единицу.
        2.1.2. Массив переписывается, без удаленной записи.

Выходные данные: скорректированный массив записей.

2.2 Алгоритмы задания по варианту

Алгоритм удаления элемента по выбранному полю

Исходные данные: массив записей
Алгоритм:
1. Выбор поля, по которому необходимо удалить запись.
2. Ввод значения.
3. Цикл, пока не будет достигнут последний элемент массива:
    2.2. Если введенное значение совпадает с данными в массиве:
        2.2.1.1. Удаление записи.
3. Вывод данных в консоль.

Выходные данные: скорректированный массив записей.

Алгоритм замены выбранного элемента

Исходные данные: массив записей
Алгоритм:
1. Ввод фамилии абитуриента, запись о котором нужно изменить.
2. Цикл, пока не будет достигнут последний элемент массива:
3. Если запись с такой фамилией найдена, то:
    2.2. Ввод новых данных.
4. Вывод данных в консоль.

Выходные данные: скорректированный массив записей.

Алгоритм удаления элементов, начиная от выбранного

Исходные данные: массив записей
Алгоритм:
1. Ввод фамилии абитуриента, начиная с которой необходимо удалить элементы.
2. Цикл, пока не будет достигнут последний элемент массива:
    2.1. Если запись с данной фамилией найдена:
        2.1.1. Удаление записей.
3. Вывод данных в консоль.

Выходные данные: скорректированный массив записей.

3 РАЗРАБОТКА ПРОГРАММЫ

3.1 Выбор средств программирования

Среда разработки: Microsoft Visual Studio 2019 Community. ОС: Windows 7. Задание выполнялось на языке программирования C++.

Для выполнения задачи потребовалось подключение стандартных библиотек iostream, Windows.h, iomanip и fstream.

#include <iostream> – подключение стандартной библиотеки, содержащая функции и переменные для организации ввода/вывода в языке программирования C++.

#include <fstream> – подключение стандартной библиотеки, предоставляющая интерфейс для чтения/записи данных из/в файл.

#include <Windows.h> – подключение библиотеки для корректного вывода записей на русском языке.

#include <iomanip> – подключение библиотеки для подключения манипуляторов (setw, left) для форматного вывода информации в текстовый файл и консоль.

Использованы функции:

int strcmp (const char *str1, const char * str2) – функция для сравнения строк;

char * strcpy_s(char * destptr, const char * srcptr) – функция для копирования строки srcptr, в строку назначения, на которую ссылается указатель destptr;

3.2 Разработка модулей

Программа разбита на 5 модулей: menu.cpp, interface.cpp, file.cpp, console.cpp, main.cpp.

В модуле menu.cpp подключаются модули interface.cpp для работы с основными операциями над данными. В модулях interface.cpp и file.cpp подключается модуль console.cpp для работы с консолью.

При разработке программы были использованы следующие типы данных: структура, объединение, перечисление, указатели, целочисленный тип int, символьный тип char, тип с плавающей точкой float, тип без значения void.

Структура – составной тип данных, в котором под одним именем объединены различные типы данных.

Объединение – составной тип данных, позволяющий размещать данные различных типов, размещаемых с учетом выравнивания в одной и той же области памяти, размер которой достаточен для хранения наибольшего элемента.

Перечисление – средство создания типа данных посредством задания ограниченного множества значений.

Указатели – переменные, в которых хранится адрес данных.

Модуль menu.cpp

Содержит функцию для организации меню.

Разработанная программа является консольным приложением, поэтому с помощью меню было организовано взаимодействие с пользователем (см. рисунок 3.1). Пользователь может выбрать необходимый пункт меню для осуществления поставленной задачи или выйти из программы.

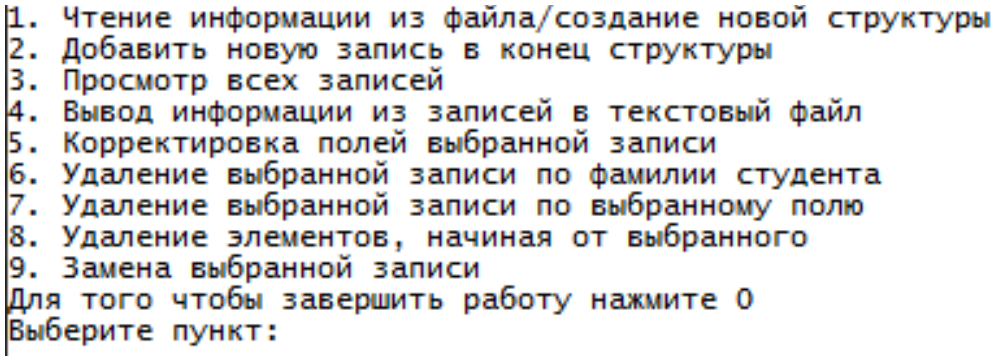

Рисунок 3.1 – Организация меню

void menu ();
Входные параметры: нет.
Назначение: организация меню.
Возвращаемые данные: нет.

Модуль interface.cpp

Содержит функции для изменения, корректировки и удаления записей, добавления записей в конец, до и после выбранной записи.

void change (int num, student* array);
Входные параметры: количество элементов, указатель на элемент массива.
Назначение: замена записи.
Возвращаемые данные: нет.

void struct_correct (int num, student* array);
Входные параметры: количество элементов, указатель на элемент массива.
Назначение: корректировка полей записи выбранного элемента.
Возвращаемые данные: нет.

void struct_delete (int &num, student* array);
Входные параметры: количество элементов, указатель на элемент массива.
Назначение: удаление выбранного элемента.
Возвращаемые данные: нет.

void deletefromchoice (int& num, student*& array);
Входные параметры: количество элементов, указатель на элемент массива.
Назначение: удаление элементов, начиная от выбранного.
Возвращаемые данные: нет.

void specdelete (int& num, student*& array);

Входные параметры: количество элементов, указатель на элемент массива.

Назначение: удаление элементов по условию (поле < или > заданного значения).

Возвращаемые данные: нет.

Модуль file.cpp

Содержит функции для работы с текстовыми файлами (ввод/запись в файл).

void filecreate (int num, student* array);

Входные параметры:

1) количество элементов;
2) указатель на элемент массива.

Назначение: если файл открыт, создание текстового файла и запись в него данных, считанных с консоли.

Возвращаемые данные: нет.

void fileread (int& num, student*& array);

Входные параметры:

1) количество элементов;
2) указатель на элемент массива.

Назначение: если файл пуст, то создание файла, создание и заполнение массива и запись массива в файл. Иначе чтение данных из текстового файла в консоль.

Возвращаемые данные: нет.

void fileadd (int num);

Входные параметры:

1) количество элементов;

Назначение: добавление новых записей в конец файла.

Возвращаемые данные: нет.

Модуль console.cpp

Содержит функции для работы с массивом данных (ввод/вывод в консоль).

void struct_create (int& num, student * & array);

Входные параметры:

1) количество элементов;
2) указатель на элемент массива.
Назначение: ввод данных с консоли в массив.
Возвращаемые данные: нет.

void struct_output (int num, student * array);
Входные параметры:
1) количество элементов;
2) указатель на элемент массива.
Назначение: вывод данных массива на консоль.
Возвращаемые данные: нет.

Модуль main.cpp

В данном модуле осуществляется смена языка консоли, вызов функции меню для взаимодействия с пользователем.

4 ТЕСТИРОВАНИЕ

4.1 Описание входных и выходных данных

При запуске программы читаются данные из файла, название которого «abiturienti.txt» (см. рисунок 4.1).

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ЭФ
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	6.2	П	ФИСЭ
7	Мэддисон	2002	4.2	Б	МСФ

Рисунок 4.1 – Содержимое файла «abiturienti.txt»

4.2 Результаты тестирования

Среда тестирование – ПК, процессор Intel Core i3-4010U с частотой 1.7 ГГц, ОЗУ 8 ГБ, тип системы: 64-разрядная ОС Windows 7.

Тест 1: «Ввод информации из текстового файла в массив указателей на записи»

Ожидаемый результат: сформированный массив записей, содержащий данные считанные из текстового файла.

Описание: тестирование правильности чтение информации из файла. Считывание происходит при выборе соответствующего пункта меню (см. рисунок 4.2).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ЭФ
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	6.2	П	ФИСЭ
7	Мэддисон	2002	4.2	Б	МСФ

Рисунок 4.2 – Данные, считанные из файла

Вывод: ввод информации из текстового файла работает корректно, массив записей, хранящий данные, считанные из файла, сформирован. Ожидаемый результат совпал с полученным.

Тест 2: «Добавление новых элементов в конец массива»

Ожидаемый результат: добавление новых элементов в конец массива.

Описание: тестирование правильности добавления новых элементов в конец массива. Добавление происходит при выборе соответствующего пункта меню. В исходный массив структурированных данных добавим записи Сидоров и Оводок (см. рисунок 4.3).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ЭФ
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	6.2	П	ФИСЭ
7	Мэддисон	2002	4.2	Б	МСФ
8	Сидоров	2001	8.5	Б	МСФ
9	Оводок	2000	10	П	ФИСЭ

Рисунок 4.3 – Данные, находящиеся в массиве

Вывод: добавление элементов в конец массива работает корректно. Ожидаемый результат совпал с полученным.

Тест 3: «Корректировка полей выбранного элемента»

Ожидаемый результат: изменение данных, хранящихся в выбранном поле выбранной записи.

Описание: тестирование правильности корректировки полей. Корректировка происходит после выбора соответствующего пункта меню, фамилии абитуриента и поля изменяемой записи. В тестовом примере изменим в записи Шевцов средний балл аттестата, в записи Оводок – форму обучения, в записи Вист – факультет (см. рисунок 4.4).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ИЭФЗО
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	3.7	П	ФИСЭ
7	Мэддисон	2002	4.2	Б	МСФ
8	Сидоров	2001	8.5	Б	МСФ
9	Оводок	2000	10	Б	ФИСЭ

Рисунок 4.4 – Массив после изменений

Вывод: корректировка полей выбранной записи работает корректно. Ожидаемый результат совпал с полученным.

Тест 4: «Удаление выбранного элемента»

Ожидаемый результат: удаление из массива выбранной записи.

Описание: тестирование правильности удаления выбранной записи. Удаление происходит после выбора соответствующего пункта меню и ввода фамилии абитуриента удаляемой записи. В тестовом примере удалим запись Мэддисон (см. рисунок 4.5).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ИЭФЗО
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	3.7	П	ФИСЭ
7	Сидоров	2001	8.5	Б	МСФ
8	Оводок	2000	10	Б	ФИСЭ

Рисунок 4.5 – Массив после удаления

Вывод: удаление выбранного элемента работает корректно. Ожидаемый результат совпал с полученным.

Тест 5: «Замена выбранного элемента»

Ожидаемый результат: замена выбранной записи.

Описание: проверка корректности замены записи. Замена записи происходит после выбора соответствующего пункта меню и ввода фамилии абитуриента. В тестовом примере заменим запись Оводок на запись Герез (см. рисунок 4.6).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ИЭФЗО
3	Левоцкий	2002	8.7	Б	ФЭИС
4	Хмурец	2001	9.5	Б	ФЭИС
5	МакГрегор	1999	8.3	П	ЭФ
6	Шевцов	2000	3.7	П	ФИСЭ
7	Сидоров	2001	8.5	Б	МСФ
8	Герез	1999	9.9	Б	ФЭИС

Рисунок 4.6 – Массив после замены записи Оводок

Вывод: замена записи работает корректно. Ожидаемый результат совпал с полученным.

Тест 6: «Удаление элементов, начиная от выбранного»

Ожидаемый результат: удаление из массива записей, начиная от выбранной и до конца.

Описание: проверка корректности удаления из массива записей, начиная от выбранной. Удаление из массива записей, начиная от выбранной происходит после выбора соответствующего пункта меню и ввода фамилии абитуриента, начиная с записи которого нужно удалить данные. В тестовом примере удалим записи, начиная с записи Хмурец (см. рисунок 4.7).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Иванюк	2003	7.9	Б	СФ
2	Вист	2002	8.9	Б	ИЭФЗО
3	Левоцкий	2002	8.7	Б	ФЭИС

Рисунок 4.7 – Массив после удаления записей

Вывод: удаление записей массива, начиная с выбранной работает корректно. Ожидаемый результат совпал с полученным.

Тест 7: «Удаление элементов по условию (поле < или > заданного значения)»

Ожидаемый результат: удаление из массива записей по заданному условию.

Описание: проверка корректности удаления из массива записей по заданному. Удаление из массива записей по заданному условию происходит после выбора соответствующего пункта меню и ввода поля, по которому необходимо удалить данные. В тестовом примере удалим запись Иванюк, выбрав поле Год рождения и введя значение 2003 (см. рисунок 4.8).

Полученный результат:

#	Фамилия:	Год рождения:	Ср. балл ат-а:	Форма обучения:	Факультет:
1	Вист	2002	8.9	Б	ИЭФЗО
2	Левоцкий	2002	8.7	Б	ФЭИС

Рисунок 4.8 – Массив после удаления записей

Вывод: удаление записи массива по выбранному полю работает корректно. Ожидаемый результат совпал с полученным.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы усвоил и закрепил знания о работе со структурами данных, основными типами данных, текстовыми файлами, динамическими массивами и указателями, и модульным программированием в языке программирования C++.

Все поставленные задачи были выполнены, а условия и ограничения соблюдены. Были разработаны алгоритмы для решения поставленных задач, обеспечено взаимодействие между модулями программы, проведено тестирование правильности выполнения реализованных функций. Все функции работают корректно, а ожидаемый результат совпадает с полученным.

В итоге была разработана программа, позволяющая хранить и оперировать информацией об абитуриентах ВУЗа, и предоставляющая для этого все необходимые инструменты. Программа является консольным приложением. С помощью меню было организовано взаимодействие с пользователем. Пользователь может выбрать необходимый пункт меню, чтобы выполнить поставленную задачу.

Данная программа может быть полезна различным ВУЗам, которым необходимо хранить и редактировать данные об абитуриентах, так как имеются все данные, которые идентифицируют абитуриента.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
4. ГОСТ 19.005-85. ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.
5. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов.
6. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
7. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов.
8. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению.
9. ГОСТ 19.402-78. ЕСПД. Описание программы.
10. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.
11. Структуры и функции // METANIT.com [Электронный ресурс]. – 2017. – Режим доступа: https: //metanit.com/cpp/c/6.5.php. – Дата доступа: 02.06.2017.
12. Функции в C++ // CODE-LIVE.com [Электронный ресурс]. – 2011. – Режим доступа: https://code-live.ru/post/cpp-functions. – Дата доступа: 02.09.2011.
13. Указатели // METANIT.com [Электронный ресурс]. – 2017. – Режим доступа: https://metanit.com/cpp/tutorial/4.1.php. – Дата доступа: 22.09.2017.