**Курсовая работа**

**по дисциплине: «Основы программирования на языке C»**

2019 г

# Задание

Разработать приложение на языке C, реализующее операции со списком студентов:

* вывод списка на печать;
* добавление нового студента;
* редактирование существующих данных студента;
* удаление записи о студенте.

Запись о студенте содержит следующую информацию:

* имя;
* возраст;
* средний балл.

Необходимо реализовать программу в виде консольного приложения Windows.

Оглавление

[Задание 2](#_Toc11019586)

[Введение 4](#_Toc11019587)

[1 Исходные данные и способ их ввода 5](#_Toc11019588)

[2 Основные модули программы 6](#_Toc11019589)

[3 Описание функций 7](#_Toc11019590)

[3.1 Функции модуля main 7](#_Toc11019591)

[3.2 Функции модуля student 7](#_Toc11019592)

[4 Описание порядка работы с программой 9](#_Toc11019593)

[Заключение 11](#_Toc11019594)

[Список литературы 12](#_Toc11019595)

[Приложение А. Исходный код программы 13](#_Toc11019596)

# Введение

В настоящее время, с развитием научно-технического прогресса и информационных технологий, сложность автоматизируемых предметных областей постоянно и неуклонно возрастает. Программист проводит всё своё рабочее время, по сути, в борьбе со сложностью программ, отлаживая свой код в поисках допущенных ошибок.

Ради борьбы со сложностью эволюционируют языки программирования. После машинных кодов и перфокарт появились текстовые редакторы и язык ассемблера, что позволило писать чуть более сложные и объёмные программы при меньших трудозатратах. Далее появились процедурные языки высокого уровня – C, Паскаль и многие другие, что перевело технологии программирования на новый уровень.

Данная работа выполнена в виде программы на языке программирования C. Программа предназначена для работы со списком студентов и позволяет добавлять, удалять и редактировать записи, а также выводить содержимое списка на экран.

Пользовательский интерфейс выполнен в виде текстовой консоли.

# Исходные данные и способ их ввода

Исходными данными в программе являются команды пользователя и информация о студентах – имя, возраст и успеваемость. Ввод данных производится пользователем с клавиатуры.

Взаимодействие с пользователем организовано в виде консольного меню. После запуска, программа печатает на экране пронумерованный список возможных действий и предлагает пользователю ввести номер команды, как показано на рисунке 1.

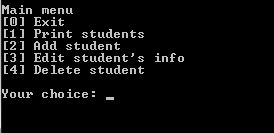


Рисунок 1. Главное меню программы

Как видно из рисунка, возможные действия пользователя – это добавление, редактирование и удаление записей о студентах, а также печать списка на экране. В зависимости от выбранного пункта, может также потребоваться ввод порядкового номера или данных студента.

# Основные модули программы

Программа состоит из 2 модулей, каждый из которых отвечает за свой функционал:

* **main** – основной модуль, управляющий текущим состоянием программы и отвечающий за организацию пользовательского ввода;
* **students** – контейнер для хранения информации о студентах, организованный в виде двусвязного списка.

Необходимость выделения отдельных модулей возникала по мере разрастания и усложнения программы, а также появления новых сущностей, наличие которых в едином файле усложняло понимание программного кода.

Например, сперва был создан основной модуль с функцией входа в программу, затем в нём был реализован пользовательский ввод. После этого уже потребовалось реализовать хранилище для вводимой информации и появился отдельный модуль, который отвечает за управление динамической структурой данных, позволяющей хранить список студентов.

# Описание функций

## Функции модуля main

**int Add\_Student(Students\* students)**

Запрашивает у пользователя информацию о студенте и добавляет новую запись в список.

**int Edit\_Student(Students\* students)**

Запрашивает у пользователя порядковый номер и новую информацию о студенте, после этого обновляет запись в списке.

**int Del\_Student(Students\* students)**

Запрашивает у пользователя порядковый номер студента и удаляет соответствующую запись из списка.

**int Read\_Student(char\* name, unsigned char\* age, float\* score)**

Вспомогательная функция, которая запрашивает у пользователя информацию о студенте.

**int Read\_Index()**

Вспомогательная функция, которая запрашивает у пользователя порядковый номер студента.

**int main(int argc, char\* argv[])**

Точка входа программы. Управляет пользовательским вводом, и выводит результат на экран.

## Функции модуля student

**void Students\_Add\_Item(Students\* list, Student\* item);**

Добавление в список нового элемента.

**void Students\_Add\_Data(Students\* list, const char\* name, unsigned char age, float score);**

Создание нового элемента с заданными параметрами и добавление его в список.

**void Students\_Edit(Student\* item, const char\* name, unsigned char age, float score);**

Редактирование элемента списка по указателю.

**void Students\_Edit\_Index(Students\* list, int index, const char\* name, unsigned char age, float score);**

Редактирование элемента списка по индексу

**void Students\_Del(Students\* list, Student\* item);**

Удаление из списка по указателю.

**void Students\_Del\_Index(Students\* list, int index);**

из списка по индексу.

**void Students\_Print(Students\* list);**

Печать списка.

**void Students\_Destroy(Students\* list);**

Удаление всего списка.

# Описание порядка работы с программой

Выбор команды производится вводом с клавиатуры её номера и нажатием клавиши Enter. Разберём каждую команду пользователя по отдельности. При попытке распечатать список в самом начале сеанса работы, программа выдаст сообщение о том, что список пуст, как показано на рисунке 2.

empty.png

Рисунок 2. Печать пустого списка

Добавление новой записи о студенте происходит следующим образом: пользователь выбирает команду №2, далее программа последовательно запрашивает ввод имени студента, его возраста и среднего балла, как показано на рисунке 3.

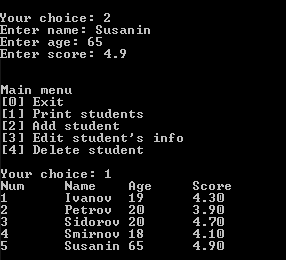


Рисунок 3. Добавление новой записи

После выполнения очередной команды на модификацию списка удобно выполнять печать его содержимого, чтобы видеть внесённые изменения.

Редактирование существующей записи о студента производится при помощи команды №3 главного меню. Последовательность действий такая же, как и при добавлении новой записи, но предварительно программа запросит порядковый номер студента, информацию о котором следует обновить, как показано на рисунке 4.

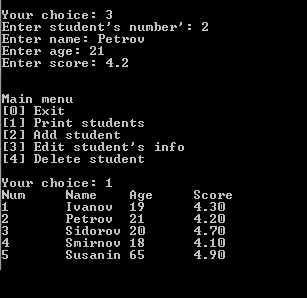


Рисунок 4. Редактирование существующей записи

Для удаления существующей записи достаточно выполнить команду №4 и ввести, в ответ на запрос программы, порядковый номер удаляемой записи, как показано на рисунке 5.

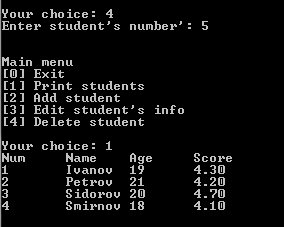


Рисунок 5. Удаление существующей записи

Пользователь может в любой момент завершить сеанс работы командой №0.

# Заключение

В результате работы была спроектирована и разработана программа, позволяющая управлять списком студентов путём добавления, удаления и редактирования информации.

В ходе работы над проектом получены практические навыки разработки программ на языке высокого уровня C. Изучено множество аспектов разработки программ на этом языке – в первую очередь, модульная организация сложной программной системы, организация динамических структур данных и управление памятью.

На базе практического применения, закреплены базовые знания, полученные ранее, при изучении теории.

# Список литературы

1. Стивен Прата Язык программирования C. Лекции и упражнения / Стивен Прата. - М.: Вильямс, 2015. - 928 c.

2. Подбельский В.В, Фомин С.С. Программирование на языке C / М.: «Финансы и статистика». 2003.

3. Дейтел Х.М., Дейтел П.Дж. Как программировать на C / М: «БИНОМ», 2000.

4. Шилдт, Г. Справочник по C. Диасофт М.: 2000.

5. ГОСТ 2.105-95 Общие требования к текстовым документам.

# Приложение А. Исходный код программы

**main.c**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include "student.h"

#define MAX\_INPUT\_LEN 100

char input[ MAX\_INPUT\_LEN ];

// Предварительные объявления функций

void Add\_Student( Students\* students );

void Del\_Student( Students\* students );

void Edit\_Student( Students\* students );

// Основная функция программы

int main( int argc, char \*argv[] ) {

int length;

//int counter = 0;

int menu = -1;

Students students;

memset( &students, 0x00, sizeof( students ) );

while ( menu != 0 ) {

printf( "\n\nMain menu\n" );

printf( "[0] Exit\n" );

printf( "[1] Print students\n" );

printf( "[2] Add student\n" );

printf( "[3] Edit student's info\n" );

printf( "[4] Delete student\n" );

printf("\n");

printf( "Your choice: " );

length = strlen( gets( input ) );

if( length == 1 && isdigit( input[ 0 ] ) ) {

menu = atoi( input );

switch ( menu ) {

case 1:

Students\_Print( &students );

break;

case 2:

Add\_Student( &students );

break;

case 3:

Edit\_Student( &students );

break;

case 4:

Del\_Student( &students );

break;

}

} else {

printf( "Not number\n" );

}

}

// освобождение ресурсов перед выходом

Students\_Destroy( &students );

printf( "Finished\n" );

return 0;

}

// Запрашивает у пользователя информацию о студенте

void Read\_Student( char\* name, unsigned char\* age, float\* score )

{

char input[ MAX\_INPUT\_LEN ];

printf( "Enter name: " );

gets( name );

printf( "Enter age: " );

gets( input );

\*age = ( unsigned char )atoi( input );

printf( "Enter score: " );

gets( input );

\*score = atof( input );

}

// Запрашивает у пользователя порядковый номер студента

int Read\_Index()

{

char input[ MAX\_INPUT\_LEN ];

printf( "Enter student's number': " );

gets( input );

return atoi( input );

}

// Добавление нового студента

void Add\_Student( Students\* students )

{

char name[ MAX\_NAME\_LENGTH ];

unsigned char age;

float score;

Read\_Student( name, &age, &score );

Students\_Add\_Data( students, name, age, score );

}

// Редактирование информации о выбранном студенте

void Edit\_Student( Students\* students )

{

char name[ MAX\_NAME\_LENGTH ];

unsigned char age;

float score;

int index = Read\_Index();

if( index > 0 ) {

Read\_Student( name, &age, &score );

Students\_Edit\_Index( students, index - 1, name, age, score );

}

}

// Удаление выбранного студента из списка

void Del\_Student( Students\* students )

{

int index = Read\_Index();

if( index > 0 ) {

Students\_Del\_Index( students, index - 1 );

}

}

**student.h**

#ifndef \_\_STUDENT\_H\_\_

#define \_\_STUDENT\_H\_\_

#define MAX\_NAME\_LENGTH 100

// Двусвязный список для хранения информации о студентах

// элемент двусвязного списка

typedef struct Student {

struct Student\* next;

struct Student\* prev;

char name[ MAX\_NAME\_LENGTH ];

unsigned char age;

float score;

} Student;

// структура двусвязного списка

typedef struct Students

{

// указатели на первый и последний элемент списка

struct Student \*head;

struct Student \*tail;

} Students;

// Добавление в список нового элемента

void Students\_Add\_Item( Students\* list, Student\* item );

// Добавление в список по переданным параметрам

void Students\_Add\_Data( Students\* list, const char\* name, unsigned char age, float score );

// Редактирование элемента списка по указателю

void Students\_Edit( Student\* item, const char\* name, unsigned char age, float score );

// Редактирование элемента списка по индексу

void Students\_Edit\_Index( Students\* list, int index, const char\* name, unsigned char age, float score );

// Удаление из списка по элементу

void Students\_Del( Students\* list, Student\* item );

// Удаление из списка по индексу

void Students\_Del\_Index( Students\* list, int index );

// Печать элемента списка

void Students\_Print( Students\* list );

// Удаление всего списка

void Students\_Destroy( Students\* list );

#endif // \_\_STUDENT\_H\_\_

**student.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "student.h"

// Добавление в список нового элемента

void Students\_Add\_Item( Students\* list, Student\* item )

{

// заносим новый элемент в список

if ( list->head == NULL ) {

list->head = list->tail = item;

item->next = item->prev = NULL;

} else {

list->tail->next = item;

item->prev = list->tail;

list->tail = item;

item->next = NULL;

}

}

// Добавление в список по переданным параметрам

void Students\_Add\_Data( Students\* list, const char\* name, unsigned char age, float score )

{

// выделяем память под новый элемент

Student \*item = ( Student\* )malloc( sizeof( Student ) );

// копируем данные

strcpy( item->name, name );

item->age = age;

item->score = score;

Students\_Add\_Item( list, item );

}

// Редактирование элемента списка по указателю

void Students\_Edit( Student\* item, const char\* name, unsigned char age, float score )

{

strcpy( item->name, name );

item->age = age;

item->score = score;

}

// Редактирование элемента списка по индексу

void Students\_Edit\_Index( Students\* list, int index, const char\* name, unsigned char age, float score )

{

int i = 0;

Student\* item = list->head;

while( item != NULL ) {

if( i++ == index ) {

Students\_Edit( item, name, age, score );

break;

}

item = item->next;

}

}

// Удаление из списка по элементу

void Students\_Del( Students\* list, Student\* item )

{

if( item->prev ) {

item->prev->next = item->next;

} else {

list->head = item->next;

}

if( item->next ) {

item->next->prev = item->prev;

} else {

list->tail = item->prev;

}

free( item );

}

// Удаление из списка по индексу

void Students\_Del\_Index( Students\* list, int index )

{

int i = 0;

Student\* item = list->head;

while( item != NULL ) {

if( i++ == index ) {

Students\_Del( list, item );

break;

}

item = item->next;

}

}

// Печать элемента списка

void Students\_Print( Students\* list )

{

int i = 0;

Student \*item = list->head;

if( item == NULL ) {

printf( "List is empty\n" );

return;

}

printf( "Num\tName\tAge\tScore\n" );

while ( item != NULL ) {

printf( "%d\t%s\t%d\t%.2f\n", ++i, item->name, (int)item->age, item->score );

item = item->next;

}

printf( "\n" );

}

// Удаление всего списка

void Students\_Destroy( Students\* list )

{

Student \*temp, \*item = list->head;

while( item != NULL ) {

temp = item;

item = item->next;

free( temp );

}

list->head = list->tail = NULL;

}