**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по курсу «технологии программирования и структуры данных»

Программирование линейных структур данных специального вида

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  | | Хромов А.М. | |
| Группа АС-17 |  | |  | |
| Руководитель |  | | Домашнев П.А. | |
| Доцент, кандидат технических наук | |  | |  |

Липецк 2018г.

# Цель работы

Приобретение навыков реализации линейных структур данных специального вида: блоковых списков, индексированных списков, индексированных блоковых списков.

Задание

Написать программу, предоставляющую возможность ведения информационного массива данных заданной структуры. Для хранения информационного массива должна использоваться линейная структура данных специального вида (в соответствие с вариантом). Программа должна реализовывать следующие функции:

* 1. Добавление новой записи в начало списка
  2. Добавление новой записи в конец списка
  3. Вставка новой записи на заданную позицию
  4. Удаление записи, находящейся на заданной позиции
  5. Изменение записи, находящейся на заданной позиции
  6. Очистка информационного массива
  7. Последовательный поиск записи в информационном массиве
  8. Вывод содержимого информационного массива на экран
  9. Вывод служебных данных и текущей структуры используемой структуры хранения.



Текст программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#include <conio.h>

#include<locale.h>

#include <math.h>

typedef struct USE

{

float points[3];

int number;

char name[20];

float sr\_ball;

int index;

} USE;

typedef struct Block

{

USE\* massiv[9];

Block\* next;

Block\* pred;

int kolvo;

} Block;

int Menu() {

int k;

printf("\nВыбор действия со списком:\n");

printf("1) Добавление элемента в начало списка;\n");

printf("2) Добавление элемента в конец списка;\n");

printf("3) Вставка нового элемента на заданную позицию\n");

printf("4) Удаление элемента с заданной позиции;\n");

printf("5) Изменение выбранной записи;\n");

printf("6) Очистка;\n");

printf("7) Поиск нужной записи;\n");

printf("8) Вывод содержимого информационного массива на экран;\n");

printf("9) Вывод служебных данных и текущей структуры используемой структуры хра-нения;\n");

printf("0) Выход;\n");

fflush(stdin);

scanf("%d", &k);

return k;

}

Block\* Create(Block\* head)

{

Block\* tmp = (Block\*)malloc(sizeof(Block));

tmp->next = NULL;

tmp->pred = NULL;

return tmp;

}

USE\* Enter()

{

USE\* tmp = (USE\*)malloc(sizeof(USE));

//printf("Введите номер студента: ");

fflush(stdin);

tmp->number = 1000;

//scanf("%d", &tmp->number);

printf("Введите имя студента: ");

fflush(stdin);

scanf("%s", tmp->name);

for (int i = 0; i < 3; i++)

{

//printf("Введите баллы студента за ЕГЭ[%d]: ", i + 1);

fflush(stdin);

tmp->points[i] = 15;

//scanf("%f", &tmp->points[i]);

}

//printf("Введите средний балл студента за ЕГЭ: ");

fflush(stdin);

tmp->sr\_ball = 20;

//scanf("%f", &tmp->sr\_ball);

return tmp;

}

Block\* Get\_By\_Index(Block\* head, int index)

{

Block\* tmp = head;

for (int i = 0; i < index; i++)

{

if (tmp->next != NULL)

{

tmp = tmp->next;

}

else return NULL;

}

return tmp;

}

int Get\_Index\_By\_Index(Block\* head, int\* num, int size)

{

int index = 0;

Block\* tmp = head;

while ((\*num) >= 0)

{

if ((\*num) >= tmp->kolvo)

{

(\*num) = (\*num) - tmp->kolvo;

}

else

{

break;

}

index++;

tmp = tmp->next;

}

return index;

}

int Get\_Index(Block\* head, int num)

{

Block\* tmp = head;

int n = num;

while (n >= 0)

{

if (n > tmp->kolvo)

{

n = n - tmp->kolvo;

}

else

{

break;

}

tmp = tmp->next;

}

return n;

}

void Add\_Elem(Block\* head, int num, int \*size)

{

Block\* tmp = head;

for (int i = 8; i > num; i--)

{

tmp->massiv[i] = tmp->massiv[i - 1];

}

tmp->massiv[num] = Enter();

(\*size)++;

tmp->kolvo++;

}

void Add\_Block(Block\* head, int num, int\* size, int\* kolvo\_blocks)

{

Block\* tmp = head;

Block\* tmp1 = (Block\*)malloc(sizeof(Block));

tmp1->next = tmp->next;

tmp->next = tmp1;

tmp1->pred = tmp;

if (tmp1->next != NULL)

{

tmp1->next->pred = tmp1;

}

tmp1->massiv[0] = tmp->massiv[8];

tmp1->kolvo = 1;

for (int i = 8; i > num; i--)

{

tmp->massiv[i] = tmp->massiv[i - 1];

}

tmp->massiv[num] = Enter();

(\*size)++;

(\*kolvo\_blocks)++;

}

void Add\_Elem\_to\_End(Block\* head, int\* size)

{

Block\* tmp = head;

tmp->massiv[tmp->kolvo] = Enter();

(\*size)++;

tmp->kolvo++;

}

void Add\_Elem\_to\_End\_and\_Add\_Block(Block\* head, int num, int\* size, int\* kolvo\_blocks)

{

Block\* tmp = head;

Block\* tmp1 = (Block\*)malloc(sizeof(Block));

tmp1->next = tmp->next;

tmp->next = tmp1;

tmp1->pred = tmp;

if (tmp1->next != NULL)

{

tmp1->next->pred = tmp1;

}

tmp1->massiv[0] = Enter();

(\*size)++;

tmp1->kolvo = 1;

(\*kolvo\_blocks)++;

}

void Del\_Elem(Block\* head, int num, int\* size)

{

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, (\*size)));

int n = Get\_Index(head, num);

free(tmp->massiv[n]);

for (int i = num; i < 8; i++)

{

tmp->massiv[i] = tmp->massiv[i + 1];

}

(\*size)--;

tmp->kolvo--;

}

void Del\_Elem\_and\_Block(Block\* head, int num, int\* size, int\* kolvo\_blocks)

{

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, (\*size)));

Block\* sl = tmp->next;

Block\* pr = tmp->pred;

free(tmp->massiv[num]);

sl->pred = pr;

pr->next = sl;

tmp->kolvo = 0;

free(tmp);

(\*size)--;

(\*kolvo\_blocks)--;

}

void Change(Block\* head, int num, int size)

{

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, size));

int n = Get\_Index(head, num);

printf("Введите номер студента: ");

fflush(stdin);

scanf("%d", &tmp->massiv[n]->number);

printf("Введите имя студента: ");

fflush(stdin);

scanf("%s", tmp->massiv[n]->name);

for (int i = 0; i < 3; i++)

{

printf("Введите баллы студента за ЕГЭ[%d]: ", i + 1);

fflush(stdin);

scanf("%f", &tmp->massiv[n]->points[i]);

}

printf("Введите средний балл студента за ЕГЭ: ");

fflush(stdin);

scanf("%f", &tmp->massiv[n]->sr\_ball);

}

void Freedom(Block\* head, int\* kolvo\_blocks, int \*size)

{

Block\* tmp = head;

Block\* tmp1 = tmp->next;

for (int i = 0; i < (\*kolvo\_blocks); i++)

{

for (int i = 0; i < tmp->kolvo; i++)

{

free(tmp->massiv[i]);

}

free(tmp);

tmp = tmp1;

}

(\*kolvo\_blocks) = 0;

(\*size) = 0;

}

void Search(Block\* head, int kolvo\_blocks)

{

char name[20];

int prover = 0;

Block\* tmp = head;

Block\* tmp1 = tmp->next;

printf("Введите имя студента, которого хотите найти: ");

fflush(stdin);

scanf("%s", &name);

for (int i = 0; i < kolvo\_blocks; i++)

{

for (int i = 0; i < tmp->kolvo; i++)

{

if (strcmp(tmp->massiv[i]->name, name) == 0)

{

printf("\n%d) %d\t%s\t", i + 1, tmp->massiv[i]->number, tmp->massiv[i]->name);

for (int t = 0; t < 3; t++)

{

printf("[%.2f]\t", tmp->massiv[i]->points[t]);

}

printf("(%.2f)\n\n", tmp->massiv[i]->sr\_ball);

prover++;

}

}

tmp = tmp1;

}

if (prover == 0)

{

printf("Элемент не найден.\n");

}

}

void Output\_to\_Display(Block\* head, int kolvo\_blocks)

{

Block\* tmp = head;

printf("--------------------------------------------------------------------------------\n");

for (int i = 0; i < kolvo\_blocks; i++)

{

for (int i = 0; i < tmp->kolvo; i++)

{

printf("%d) %d\t%s\t", i + 1, tmp->massiv[i]->number, tmp->massiv[i]->name);

for (int t = 0; t < 3; t++)

{

printf("[%.2f] ", tmp->massiv[i]->points[t]);

}

printf("\t(%.2f)\n", tmp->massiv[i]->sr\_ball);

}

tmp = tmp->next;

printf("\n");

}

printf("--------------------------------------------------------------------------------\n");

}

void Output\_to\_Display\_Blocks(Block\* head, int size, int kolvo\_blocks)

{

Block\* tmp = head;

printf("--------------------------------------------------------------------------------\n");

for (int i = 0; i < kolvo\_blocks; i++)

{

printf("Блок №%d\n", i + 1);

for (int i = 0; i < tmp->kolvo; i++)

{

printf("%d) %d\t%s\t", i + 1, tmp->massiv[i]->number, tmp->massiv[i]->name);

for (int t = 0; t < 3; t++)

{

printf("[%.2f] ", tmp->massiv[i]->points[t]);

}

printf("\t(%.2f)\n", tmp->massiv[i]->sr\_ball);

}

printf("Количество элементов в блоке: %d\n", tmp->kolvo);

tmp = tmp->next;

printf("\n");

}

printf("\nКоличество элементов: %d\n", size);

printf("Количество блоков: %d\n", kolvo\_blocks);

printf("--------------------------------------------------------------------------------\n");

}

void main() {

Block\* head = NULL;

setlocale(0, "");

int k, size = 0, index = 0, num, kolvo\_blocks = 0;

do {

k = Menu();

if (k == 1) //Добавление в начало

{

if (size == 0)

{

head = Create(head);

head->massiv[0] = Enter();

size++;

head->kolvo = 1;

kolvo\_blocks++;

}

else if (head->kolvo < 9)

{

Add\_Elem(head, 0, &size);

}

else if (head->kolvo == 9)

{

Add\_Block(head, 0, &size, &kolvo\_blocks);

}

}

else if (k == 2) //Добавление в конец

{

num = size - 1;

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, size));

if (size == 0)

{

head = Create(head);

head->massiv[0] = Enter();

size++;

head->kolvo = 1;

kolvo\_blocks++;

}

else if (tmp->kolvo < 9)

{

Add\_Elem\_to\_End(tmp, &size);

}

else if (tmp->kolvo == 9)

{

Add\_Elem\_to\_End\_and\_Add\_Block(tmp, num, &size, &kolvo\_blocks);

}

}

else if (k == 3) //Добавление на позицию

{

if (size == 0)

{

head = Create(head);

head->massiv[0] = Enter();

size++;

head->kolvo = 1;

kolvo\_blocks++;

}

else

{

printf("Введите индекс элемента, на место которого хотите вставить элемент: ");

fflush(stdin);

scanf("%d", &num);

if (num <= size)

{

if (num < 9 && num >= 0)

{

if (head->kolvo < 9)

{

Add\_Elem(head, num, &size);

}

else if (head->kolvo == 9)

{

Add\_Block(head, num, &size, &kolvo\_blocks);

}

}

else

{

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, size));

if (tmp->kolvo < 9)

{

Add\_Elem(tmp, num, &size);

}

else if (tmp->kolvo == 9)

{

Add\_Block(tmp, num, &size, &kolvo\_blocks);

}

}

}

else {

printf("Ошибка.\n");

}

}

}

else if (k == 4) //Удалить с позиции

{

printf("Введите индекс элемента, который хотите удалить: ");

fflush(stdin);

scanf("%d", &num);

Block\* tmp = Get\_By\_Index(head, Get\_Index\_By\_Index(head, &num, size));

if (tmp->kolvo == 1) {

Del\_Elem\_and\_Block(tmp, num, &size, &kolvo\_blocks);

}

else if (tmp->kolvo > 1) {

Del\_Elem(tmp, num, &size);

}

}

else if (k == 5) //Изменить запись {

printf("Введите номер элемента, который хотите изменить: ");

fflush(stdin);

scanf("%d", &num);

if (num <= size) {

Change(head, num, size);

}

else printf("Элемент не найден.\n");

}

else if (k == 6) //Очистить

{

if (size != 0)

{

Freedom(head, &kolvo\_blocks, &size);

}

}

else if (k == 7) //Поиск

{

Search(head, kolvo\_blocks);

}

else if (k == 8) //Вывод данных {

if (size > 0) {

Output\_to\_Display(head, kolvo\_blocks);

}

else printf("\nСписок пуст.\n");

}

else if (k == 9) //Вывод дополнителной информации {

Output\_to\_Display\_Blocks(head, size, kolvo\_blocks);

}

else if (k == 0) { break; } //Выход

else printf("Выбрано некорректное действие, попробуйте снова.\n\n");

} while (k != 0);

if (size != 0) {

Freedom(head, &kolvo\_blocks, &size);

}

fflush(stdin);

getchar();

}