МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий Государственный Технический Университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по технологиям программирования №3

“ Программирование линейных структур данных специального вида ”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С.М.

(подпись, дата)

Группа АС-21-1

Руководитель

Доцент, кандидат наук \_\_\_\_\_\_\_\_\_\_\_\_ Хромов А. М.

(подпись, дата)

Липецк 2022 г.

**Содержание:**

1. Титульный лист.

2. Задание кафедры, соответствующее варианту, номер варианта.

3. Цель работы.

4. Блок-схему алгоритма разработанной программы

5. Текст программы

6. Контрольный пример

7. Выводы по работе.

## 2. Задание кафедры

Вариант 18. 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:

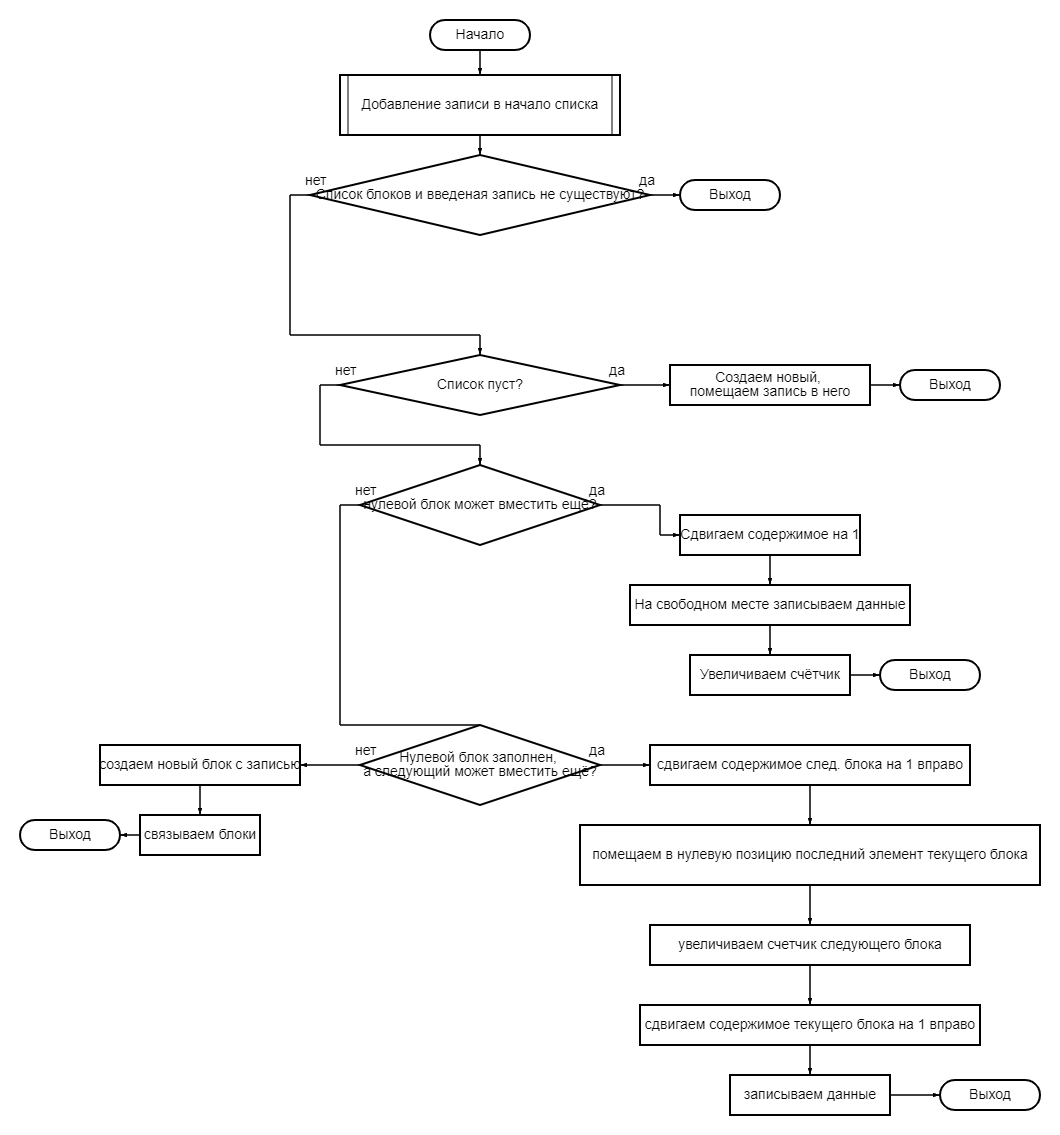
* 1. Добавление новой записи в начало списка
  2. Добавление новой записи в конец списка
  3. Вставка новой записи на заданную позицию
  4. Удаление записи, находящейся на заданной позиции
  5. Изменение записи, находящейся на заданной позиции
  6. Очистка информационного массива
  7. Последовательный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

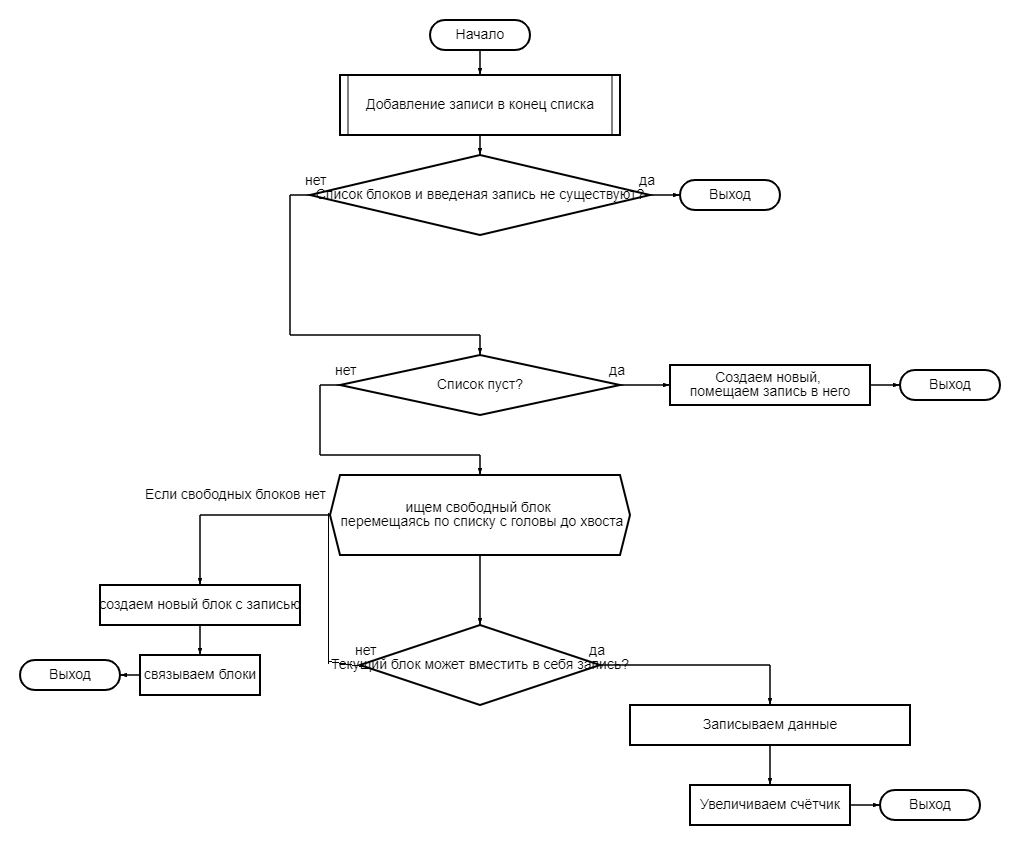
## 3. Цель работы

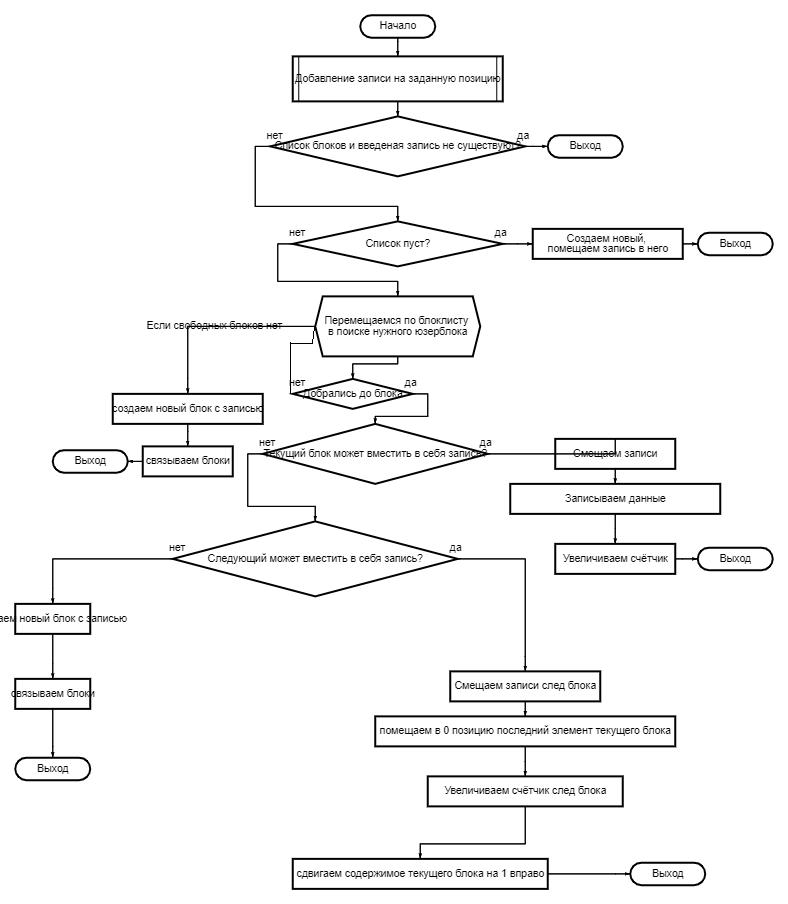
Приобретение навыков реализации линейных структур данных специального вида: блоковых списков, индексированных списков, индексированных блоковых списков.

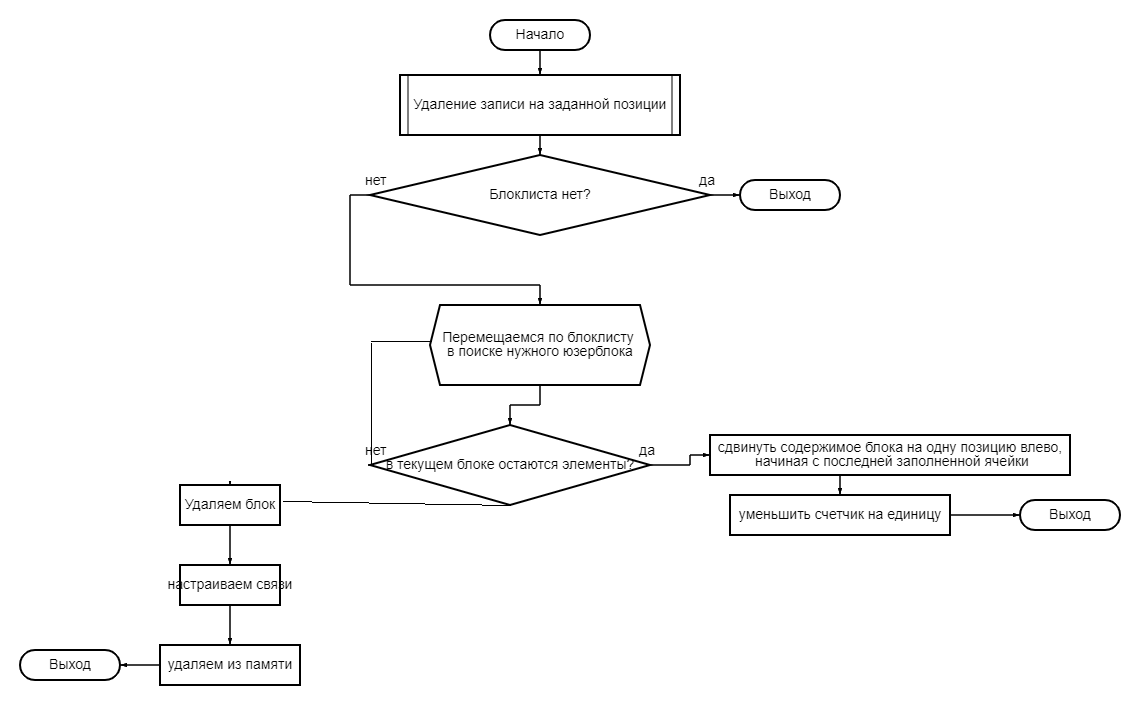
**4. Блок-схема**

1)Добавление новой записи в начало списка

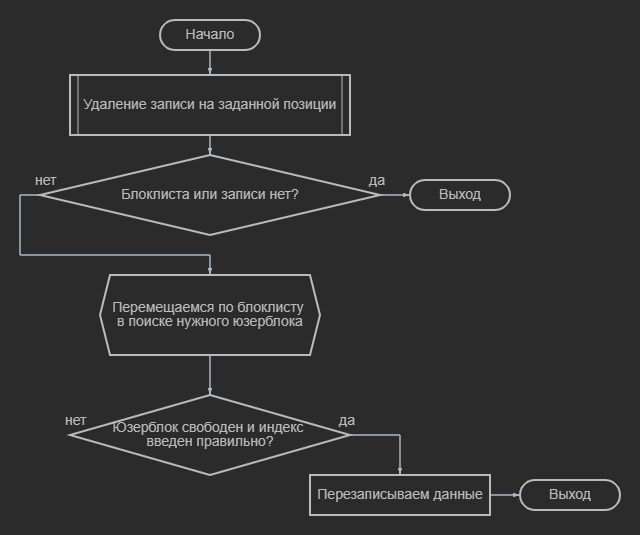
****

2) Добавление записи в конец списка****

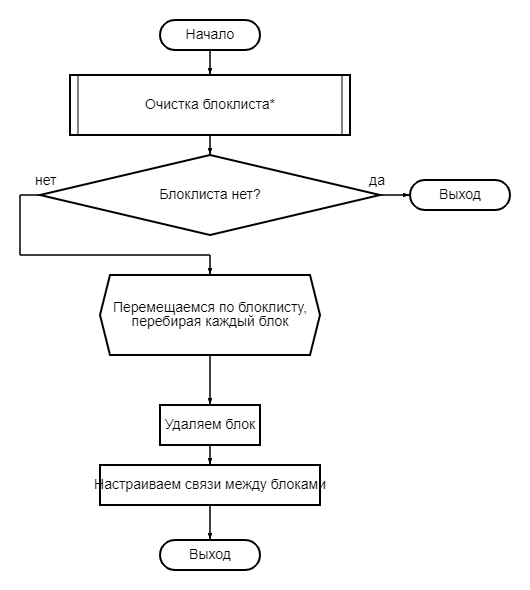
3) Вставка записи на заданную позицию****

4) Удаление записи на заданной позиции ****

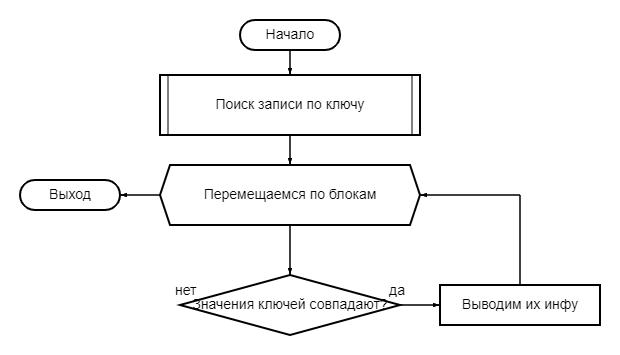
5) Изменение записи на заданной позиции



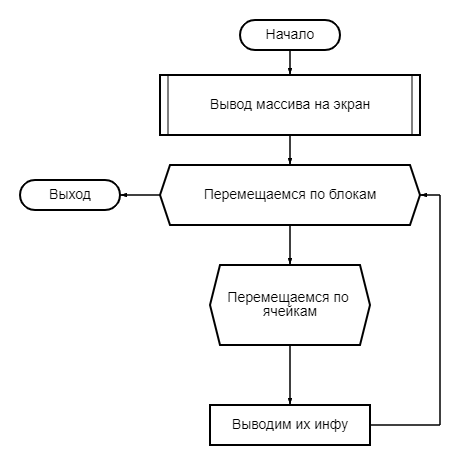
6) Очистка списка блоков

****

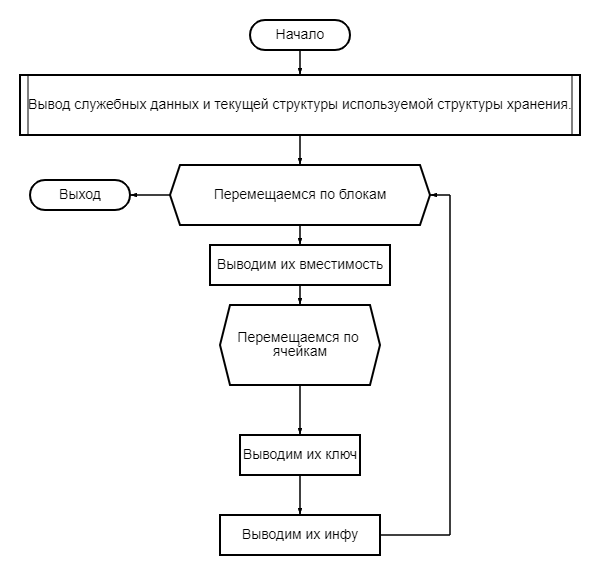
7) Поиск записи по ключу



8) Вывод массива данных на экран



9) Вывод служебных данных и текущей структуры используемой структуры хранения



**5. Код программы**

#include <iostream>

#include <vector>

using namespace std;

const size\_t blockSize = 9;

struct record //запись с данными

{

float key; //ключ

vector<int> intData; //должен быть массивом

vector<char> charData; // должен быть массивом

float floatData; //инфа

};

struct block //блок

{

size\_t cnt; //кол-во записей в блоке

record cells[blockSize]; //ячейки блока

block\* prev = nullptr; //предыдущий элемент

block\* next = nullptr; //следующий элемент

};

struct blockList //список

{

block\* head = nullptr; //указатель на голову списка

block\* tail = nullptr; //указатель на хвост списка

};

void blockList\_add\_begin(blockList\* l, record\* r) //добавляет запись в начало списка

{

if (!l || !r)

return;

//Если список пуст (указатель на голову == nullptr)

if (!l->head) {

//Создаем новый блок c этой записью и задаем head

l->head = new block{ 1, \*r };

//также задаем tail, т.к. это один блок

l->tail = l->head;

//выходим

return;

}

//Если сюда попали, значит список не пуст

//Если блок в начале (head) может вместить данные

if (l->head->cnt < blockSize)

{

//сдвигаем содержимое блоков на 1, освобождая 0 место

for (size\_t i = l->head->cnt; i > 0; i--)

{

l->head->cells[i] = l->head->cells[i - 1];

}

//на освободившееся записываем данные

l->head->cells[0] = \*r;

//увеличиваем счетчик записей

l->head->cnt++;

//выходим

return;

}

else if (l->head->cnt >= blockSize && l->head->next->cnt < blockSize)

{

for (int i = l->head->next->cnt; i > 0; i--)

{

l->head->next->cells[i] = l->head->next->cells[i - 1]; //сдвигаем содержимое след. блока на 1 вправо

}

l->head->next->cells[0] = l->head->cells[blockSize - 1]; // помещаем в нулевую позицию последний элемент текущего блока

l->head->next->cnt++; //увеличиваем счетчик следующего блока

for (int j = blockSize-1; j > 0; j--)

{

l->head->cells[j] = l->head->cells[j - 1]; //сдвигаем содержимое текущего блока на 1 вправо

}

l->head->cells[0] = \*r;

return;

}

else

{

//Если сюда попали, значит список не пуст и нет мест в head

//создаем новый блок с записью

block\* p = new block{ 1, \*r };

//связываем блоки

p->next = l->head;

l->head = p;

return;

}

}

void blockList\_add\_end(blockList\* l, record\* r)//добавляет запись в конец списка

{

//если l или r == nullptr

if (!l || !r)

//выходим

return;

//если список пуст (указатель на голову == nullptr)

if (!l->head)

{

//создаем новый блок и задаем head

l->head = new block{ 1, \*r };

//также задаем tail, т.к. это один блок

l->tail = l->head;

//выходим

return;

}

//если попали сюда, значит список не пуст и ищем свободный блок

//перемещаясь по списку с головы до хвоста

for (block\* currentBlock = l->head; currentBlock; currentBlock = currentBlock->next) {

//если текущий блок может вместить в себя запись

if (currentBlock->cnt < blockSize) {

//добавляем её

currentBlock->cells[currentBlock->cnt] = \*r;

//увеличиваем счетчик записей

currentBlock->cnt++;

//выход

return;

}

}

//если попали сюда, значит свободных блоков не было, надо добавить новый

block\* p = new block{ 1, \*r };

//у последнего элемента next связываем с новым блоком

l->tail->next = p;

//связываем новый элемент с предыдущим последним

p->prev = l->tail;

//передвигаем хвост на созданный блок

l->tail = p;

}

void blockList\_add\_custom(blockList\* l, record\* r, size\_t indexBlock, const size\_t indexCell) //Удаление строчки

{

//если l или r == nullptr

if (!l || !r)

//выходим

return;

block\* temp = l->head;

//Если список пуст (указатель на голову == nullptr)

if (!temp)

{

//Создаем новый блок c этой записью и задаем head

temp = new block{ 1, \*r };

//выходим

return;

}

/\*считаем блоки, не доходя до последнего.

Если это последний блок, проверяем не превышало значение indexBlock фактическому

количеству блоков. Если нет, тогда добавляем из массива записей этого блока переданные в функцию

номер блока->номер строчки, при условии, что indexCell < blockSize\*/

while (indexBlock != 0 && temp != l->tail) // перемещаемся по блоклисту в поиске нужного юзерблока

{

temp = temp->next;

indexBlock--;

}

if (indexBlock == 0 && indexCell < blockSize) // Когда нашли

{

if (temp->cnt < blockSize) // Есть свободные слоты?

{

for (size\_t i = temp->cnt; i > indexCell; i--) // смещаем

{

temp->cells[i] = temp->cells[i - 1];

}

//на освободившееся место записываем данные

temp->cells[indexCell] = \*r;

//увеличиваем счетчик записей

temp->cnt++;

//выходим

return;

}

else if (temp->cnt >= blockSize && temp->next->cnt < blockSize)

{

for (int i = temp->next->cnt; i > 0; i--)

{

temp->next->cells[i] = temp->next->cells[i - 1]; //сдвигаем содержимое след. блока на 1 вправо

}

temp->next->cells[0] = temp->cells[blockSize - 1]; // помещаем в 0 позицию последний элемент текущего блока

temp->next->cnt++; //увеличиваем счетчик следующего блока

for (int j = blockSize - 1; j > indexCell; j--)

{

temp->cells[j] = temp->cells[j - 1]; //сдвигаем содержимое текущего блока на 1 вправо

}

temp->cells[indexCell] = \*r;

return;

}

else

{

//Если сюда попали, значит список не пуст и нет мест в temp[indexBlock]

//создаем новый блок с записью

block\* k = new block{ 1, \*r };

//связываем блоки

k->next = temp->next;

temp->next = k;

k->prev = temp;

return;

}

}

}

void blockList\_edit\_custom(blockList\* l, const record\* r, size\_t indexBlock, const size\_t indexCell) //Изменение строчки

{

//если l или r == nullptr

if (!l || !r)

//выходим

return;

block\* temp = l->head;

/\*считаем блоки, не доходя до последнего. Если это последний блок, проверяем не превышало значение indexBlock фактическому количеству блоков. Если нет, тогда добавляем в массив записей этого блока переданную в функцию запись, при условии, что indexCell < blockSize\*/

while (indexBlock != 0 && temp != l->tail)

{

temp = temp->next;

indexBlock--;

}

if (indexBlock == 0 && indexCell < blockSize)

{

if (temp->cnt <= blockSize)

{

//на указанное место перезаписываем данные

temp->cells[indexCell] = \*r;

//выходим

return;

}

}

}

void blockList\_delete\_custom(blockList\* l, size\_t indexBlock, const size\_t indexCell) //Удаление строчки

{

//если l или r == nullptr

if (!l)

//выходим

return;

block\* temp = l->head;

/\*считаем блоки, не доходя до последнего.

Если это последний блок, проверяем не превышало значение indexBlock фактическому

количеству блоков. Если нет, тогда удаляем из массива записей этого блока переданные в функцию

номер блока->номер строчки, при условии, что indexCell < blockSize\*/

while (indexBlock != 0 && temp != l->tail) // перемещаемся по блоклисту в поиске нужного юзерблока

{

temp = temp->next;

indexBlock--;

}

if (temp->cnt > 1) { // если в текущем блоке остаются элементы (т.е. cnt > 1), то сдвинуть содержимое

//блока на одну позицию влево, начиная с последней заполненной ячейки,

//до позиции удаляемого элемента; уменьшить счетчик на единицу

{

for (size\_t i = indexCell+1; i < temp->cnt; i++) // Сдвигаем влево

{

temp->cells[i-1] = temp->cells[i];

}

//Удаляем ячейку

temp->cnt--;

//выходим

return;

}

}

if (temp->cnt <= 1) { //исключить опустевший блок полностью, удалив узел в двусвязном списке, настроив указатели previous и next, где необходимо

//связываем указ. предыдущего элемента, указывающий на следующий с указателем на следующим за temp

if (temp->prev)

temp->prev->next = temp->next;

//если за удаляемым ещё есть элемент - связываем его prev с предыдущим temp

if (temp->next)

temp->next->prev = temp->prev;

if (temp == l->head)

l->head = temp->next;

if (temp == l->tail)

l->tail = temp->prev;

//Удаляем из памяти

delete temp;

return;

}

}

void Search\_record(blockList\* l, record\* r)

{

for (block\* currentBlock = l->head; currentBlock; currentBlock = currentBlock->next) // перемещаемся по блокам

{

for (size\_t j = 0; j < currentBlock->cnt; j++)

{

//если значения ключей совпадают

if (r->key == currentBlock->cells[j].key)

{

cout << "Result of searching: ";

//выводим их инфу

for (size\_t i = 0; i < currentBlock->cells[j].intData.size(); i++)

{

cout << currentBlock->cells[j].intData[i] << " ";

}

for (size\_t i = 0; i < currentBlock->cells[j].charData.size(); i++)

{

cout << currentBlock->cells[j].charData[i] << " ";

}

cout << currentBlock->cells[j].floatData << endl;

}

}

}

}

void record\_print(record\* r) { // Функция перебора каждого значения в блоке + вывод

if (!r)

return;

cout <<

"{" << endl

<< " key\t:" << r->key << endl

<< " data\t:";

for (size\_t i = 0; i < r->intData.size(); i++)

{

cout << r->intData[i] << " ";

}

for (size\_t i = 0; i < r->charData.size(); i++)

{

cout << r->charData[i] << " ";

}

cout << r->floatData

<< endl

<< "}" << endl;

}

void blockList\_printItem(block\* p) { // Вызывает функцию вывода содержимого блока + его вместимость

if (!p)

return;

cout << "block (capacity:" << blockSize - p->cnt << ")\n";

for (size\_t i = 0; i < p->cnt; i++)

{

record\_print(&p->cells[i]);

}

}

void blockList\_print(blockList\* l) //Функция пробегается по всем блокам и применяет к ним предыдущую функцию

{

if (!l)

return;

for (block\* currentBlock = l->head; currentBlock; currentBlock = currentBlock->next)

{

blockList\_printItem(currentBlock);

}

}

void record\_Just\_print(record\* r) { // Функция перебора каждой cells в блоке + вывод

if (!r)

return;

cout << r->key << " ";

for (size\_t i = 0; i < r->intData.size(); i++)

{

cout << r->intData[i] << " ";

}

for (size\_t i = 0; i < r->charData.size(); i++)

{

cout << r->charData[i] << " ";

}

cout << r->floatData

<< endl;

}

void blockList\_Just\_printItem(block\* p) { // Вызывает функцию вывода содержимого блока + проход по всему блоку

if (!p)

return;

for (size\_t i = 0; i < p->cnt; i++)

{

record\_Just\_print(&p->cells[i]);

}

}

void blockList\_Just\_Print(blockList\* l)

{

if (!l)

return;

for (block\* currentBlock = l->head; currentBlock; currentBlock = currentBlock->next) {

blockList\_Just\_printItem(currentBlock);

}

}

void blockList\_clear(blockList\* l)

{

if (!l)

return;

for (block\* currentBlock = l->head; currentBlock;)

{

block\* tmp = currentBlock;

currentBlock = currentBlock->next;

delete tmp;

}

l->head = nullptr;

l->tail = nullptr;

}

int main()

{

blockList l;

int N, n;

char P;

int K;

cout << "---Main menu---" << endl

<< "1. Input to the start" << endl

<< "2. input to the end" << endl

<< "3. Input to the custom position" << endl

<< "4. Delete the custom position" << endl

<< "5. Edit the custom position" << endl

<< "6. Clear block list" << endl

<< "7. Search by the key" << endl

<< "8. Output info" << endl

<< "9. Output better info" << endl

<< "10. Exit" << endl;

while (1)

{

cin >> N;

if (N == 1)

{

cout << "1. Input to the start:" << endl;

record r;

cout << "Input key: ";

cin >> r.key;

cout << "input intData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> K;

r.intData.push\_back(K);

}

char P;

cout << "input charData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> P;

r.charData.push\_back(P);

}

cout << "input floatData: ";

cin >> r.floatData;

blockList\_add\_begin(&l, &r);

}

if (N == 2)

{

cout << "2. Input to the end:" << endl;

record r;

cout << "Input key: ";

cin >> r.key;

cout << "input intData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> K;

r.intData.push\_back(K);

}

char P;

cout << "input charData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> P;

r.charData.push\_back(P);

}

cout << "input floatData: ";

cin >> r.floatData;

blockList\_add\_end(&l, &r);

}

if (N == 3)

{

size\_t indexBlock, indexCell; record r;

cout << "3. Input to the custom position:" << endl;

cout << "Input index of block:"; cin >> indexBlock;

cout << "Input index of cell:"; cin >> indexCell;

cout << "Input key: ";

cin >> r.key;

cout << "input intData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> K;

r.intData.push\_back(K);

}

char P;

cout << "input charData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> P;

r.charData.push\_back(P);

}

cout << "input floatData: ";

cin >> r.floatData;

blockList\_add\_custom(&l, &r, indexBlock, indexCell);

}

if (N == 4)

{

size\_t indexBlock, indexCell;

cout << "4. Delete the custom position :" << endl;

cout << "Input index of block:"; cin >> indexBlock;

cout << "Input index of cell:"; cin >> indexCell;

blockList\_delete\_custom(&l, indexBlock, indexCell);

}

if (N == 5)

{

size\_t indexBlock, indexCell; record r;

cout << "5. Edit the custom position:" << endl;

cout << "Input index of block: "; cin >> indexBlock;

cout << "Input index of cell: "; cin >> indexCell;

cout << "Input key: ";

cin >> r.key;

cout << "input intData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> K;

r.intData.push\_back(K);

}

char P;

cout << "input charData: ";

cin >> n;

for (int i = 0; i < n; i++) //один способ

{

cin >> P;

r.charData.push\_back(P);

}

cout << "input floatData: ";

cin >> r.floatData;

blockList\_edit\_custom(&l, &r, indexBlock, indexCell);

}

if (N == 6)

{

blockList\_clear(&l);

}

if (N == 7)

{

cout << "7. Search by the key: " << endl;

record r;

cout << "Input key: ";

cin >> r.key;

Search\_record(&l, &r);

}

if (N == 8)

{

blockList\_Just\_Print(&l);

}

if (N == 9)

{

blockList\_print(&l);

}

if (N == 10)

{

cout << "10. Exit";

break;

}

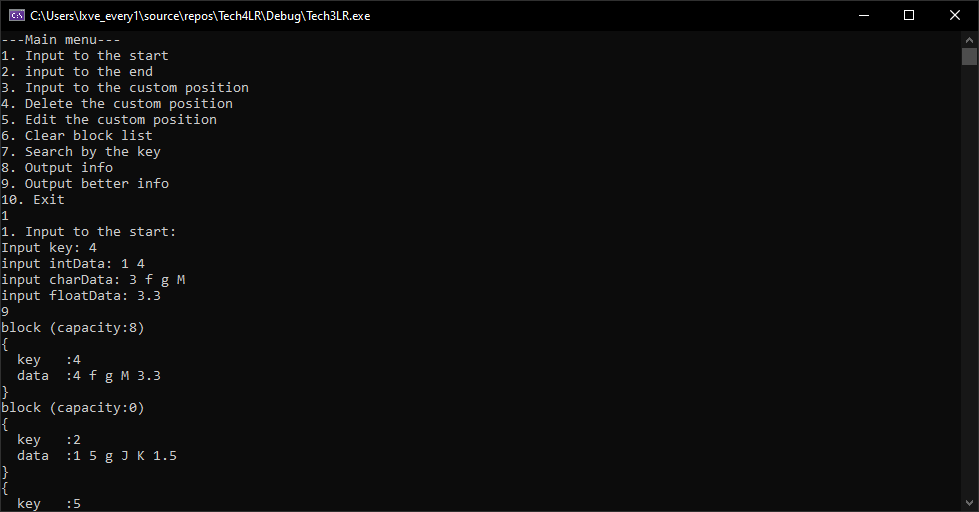
}

return 0;

}

**6. Контрольный пример**

1) Добавим дополнительно 20 записей, запустим программу, выберем первую функцию, введем данные новой записи, сразу же проверим её место в списке (функцией 9)



Как видим, из-за переполнения блоков запись отправилась в новый блок и стала первой в списке.

2) Введя данные, увидим, что в результате наша запись попала в конец списка



3) Введя данные, увидим, что в результате наша запись не нашла свободную ячейку в переполненных блоках и поместилась в новый блок [1][0]



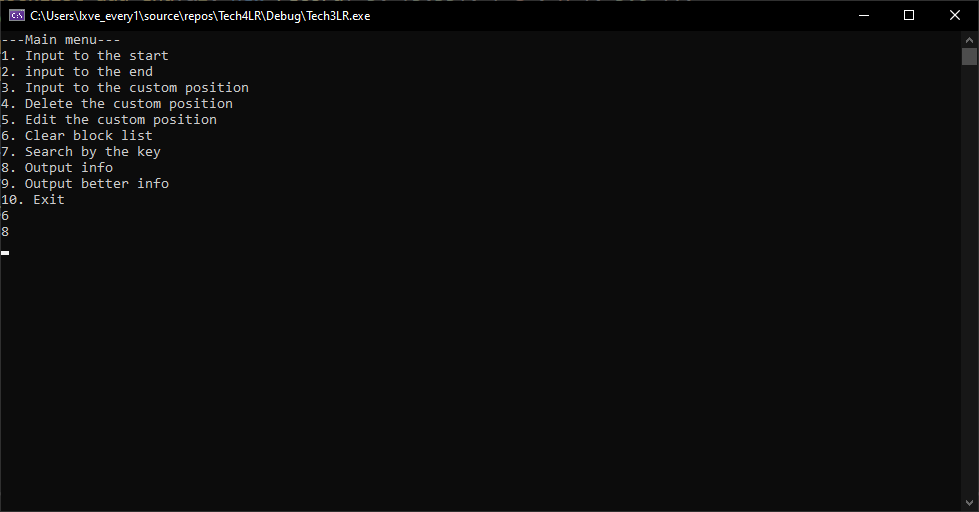
4) Введя индекс блока и ячейки заметим, что запись { 2, {1, 5}, {'g', 'J', 'K'}, 1.5} уже там не находится, а кол-во свободных ячеек увеличилось на 1.



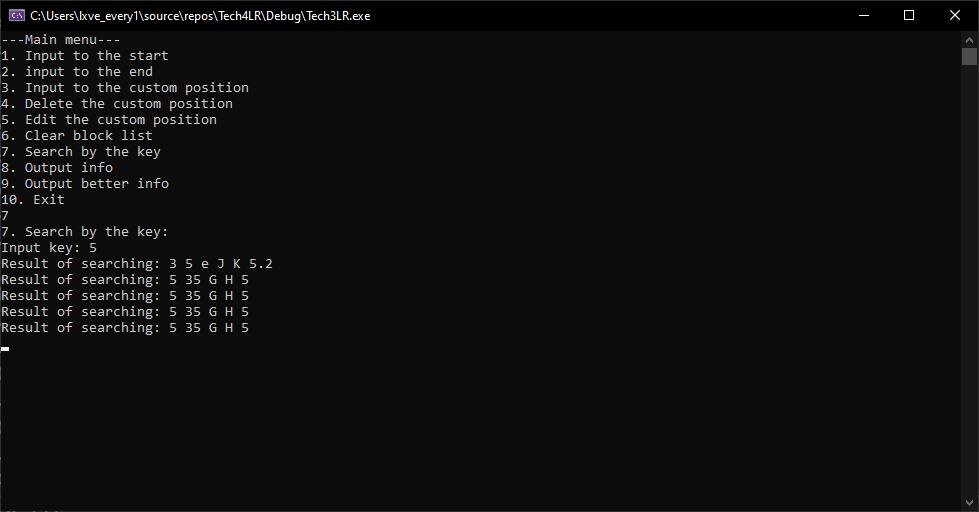
5) Введя нужные данные, заметим, что запись действительно изменилась.



6) Применим эту функцию, убедимся, что наш блок лист очистился



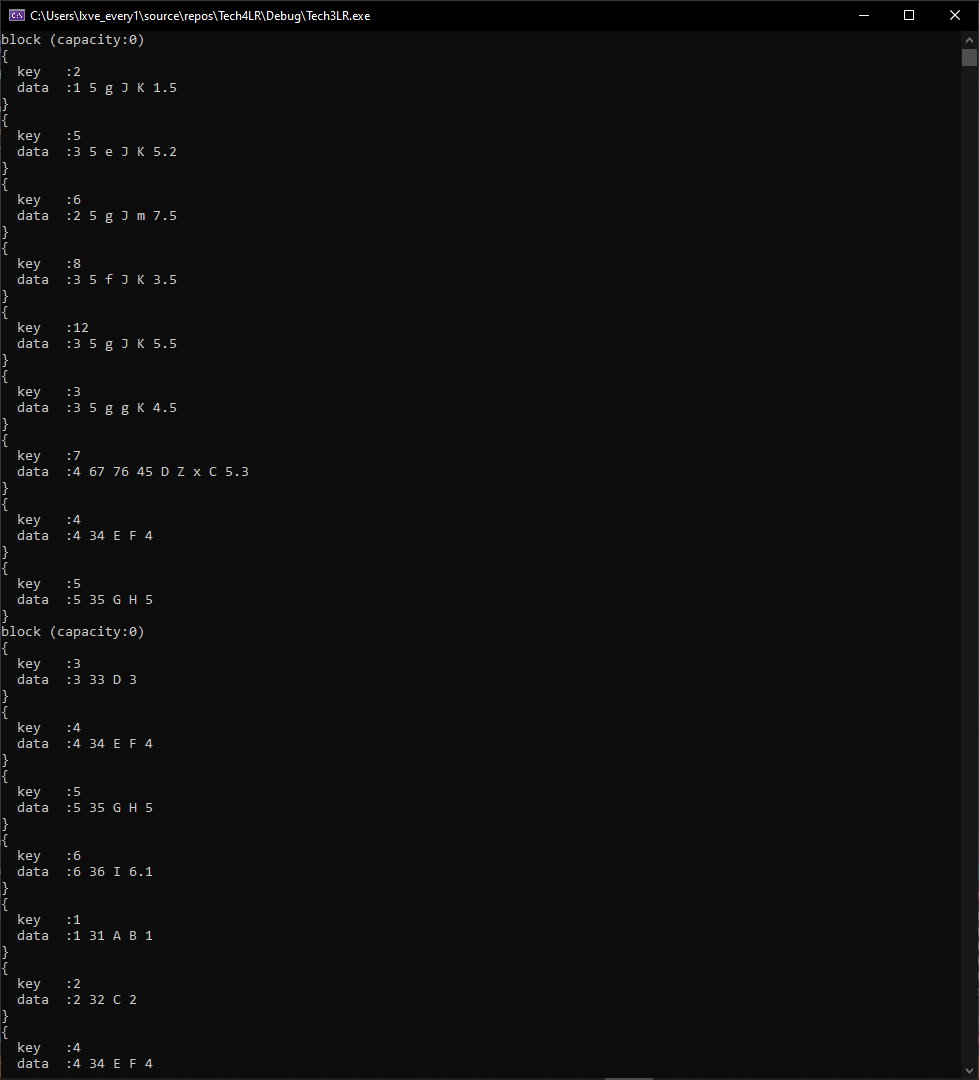
7) Введем нужный нам ключ – 5. Действительно, в нашем списке 5 ключей с соответствующим ключом



8) Вывод содержимого списка



9) Более детальный вывод содержимого списка

**7. Вывод**

В ходе выполненной работы приобрел навыки реализации линейных структур данных - блоковых списков.