Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОЛАТОРНОЙ РАБОТЕ №11

Тема: «Информационные динамические структуры»

Выполнил

Студент РИС-22-2б

Радостев И.А.

Проверил

Доцент кафедры ИТАС

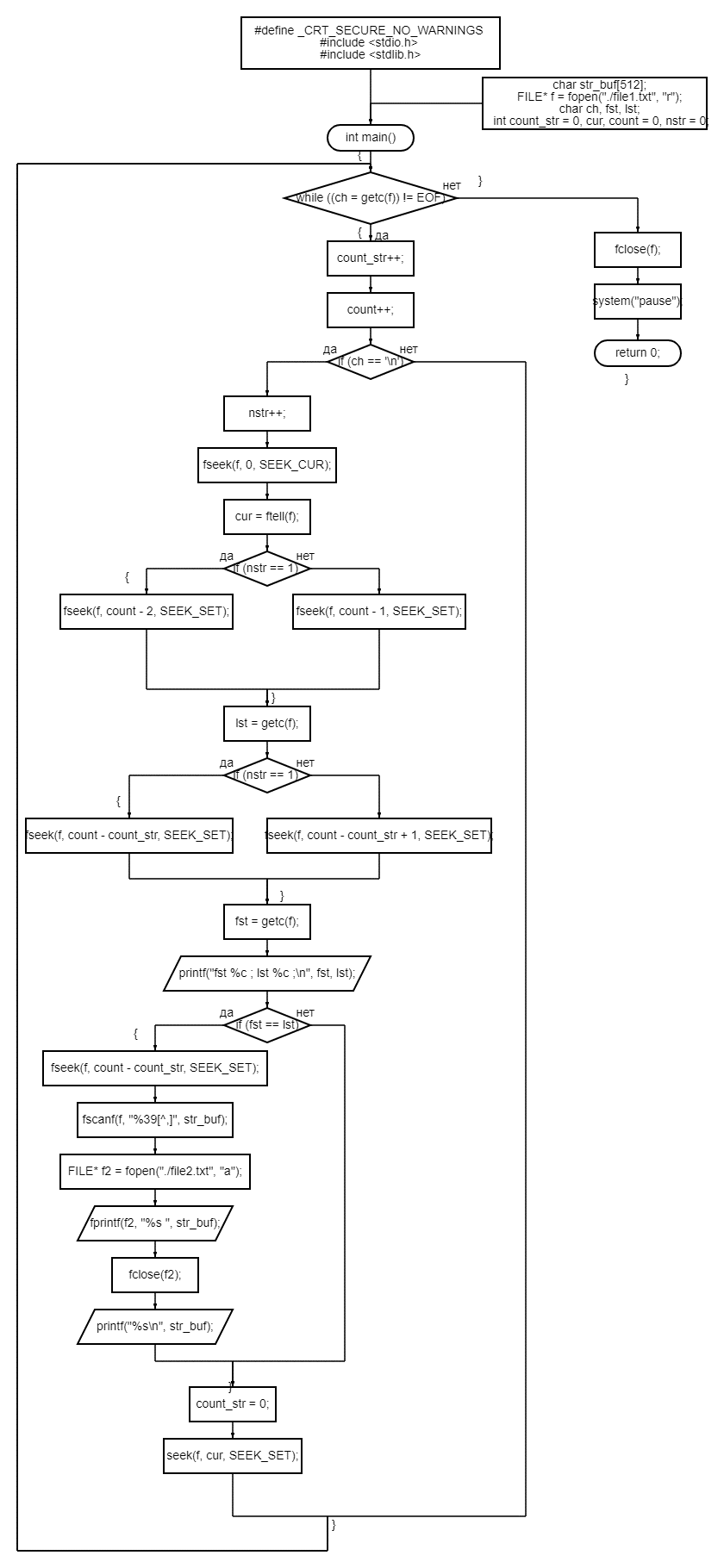
Полякова О. А.

Пермь 2023

**Постановка задачи**

Записи в линейном списке содержат ключевое поле типа int. Сформировать однонаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом.

**Блок – схема**



**Коды программ**

struct list

{

struct list\* next;

int info;

};

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <windows.h>

int input\_info(struct list\* ptr)

{

fflush(stdin);

scanf("%d", &(ptr->info));

return ptr->info;

}

struct list\* printOn(struct list\* ptr)

{

printf("%d ", ptr->info);

return (ptr->next);

}

struct list\* display(struct list\* head)

{

struct list\* ret;

if (head == NULL)

{

puts("Список пуст");

return NULL;

}

do

{

printOn(head);

ret = head;

head = head->next;

} while (head != NULL);

return (ret);

}

void destroy(struct list\* element)

{

free(element);

}

int hasMember(struct list\* head, struct list\* work)

{

while (head != NULL)

{

if (head->info == work->info) return 1;

head = head->next;

}

return(0);

}

int add(struct list\*\* head, struct list\* new\_ptr)

{

struct list\* first, \* second;

if ((\*head) == NULL)

{

(\*head) = new\_ptr;

new\_ptr->next = NULL;

return 0;

}

if ((\*head)->next == NULL)

{

if ((\*head)->info > new\_ptr->info)

{

second = (\*head);

(\*head) = new\_ptr;

new\_ptr->next = second;

second->next = NULL;

}

else

{

(\*head)->next = new\_ptr;

new\_ptr->next = NULL;

}

return 1;

}

else {

if ((\*head)->info > new\_ptr->info)

{

second = (\*head);

(\*head) = new\_ptr;

new\_ptr->next = second;

return 4;

}

first = (\*head);

second = first->next;

while (first->next != NULL)

{

if (first->info <= new\_ptr->info &&second->info >= new\_ptr->info)

{

first->next = new\_ptr;

new\_ptr->next = second;

return 2;

}

first = second;

second = first->next;

}

first->next = new\_ptr;

new\_ptr->next = NULL;

return 3;

}

}

void detach(struct list\*\* head, struct list\* element)

{

struct list\* prev;

if (\*head == NULL) return;

if ((\*head) == element)

{

(\*head) = element->next;

destroy(element);

return;

}

prev = (\*head);

while ((prev->next) != element)

{

prev = prev->next;

}

prev->next = element->next;

destroy(element);

return;

}

int main(void) {

char ch;

struct list\* head;

struct list\* new\_ptr;

struct list\* cur;

struct list work;

head = NULL;

puts("Input the numbers, one in one line, 0 - end of list");

while (input\_info(&work) != 0)

{

if (hasMember(head, &work) != 1)

{

if ((new\_ptr = (struct list\*)malloc(sizeof(struct list))) == NULL)

{

puts("No memory!");

break;

}

new\_ptr->info = work.info;

add(&head, new\_ptr);

}

}

if (head == NULL)

{

return 1;

}

puts("\*\*\* ALL LIST \*\*\*");

display(head);

cur = head;

puts("\*\*\* DELETING \*\*\*");

do

{

new\_ptr = printOn(cur);

printf("Delete the item ? (Y/N) ");

if ((ch = getch()) == 'Y' || ch == 'y')

{

puts("Yes");

detach(&head, cur);

}

else puts("No");

cur = new\_ptr;

}

while (new\_ptr != NULL);

{

puts("\*\*\* ALL LIST AFTER DELETING \*\*\*");

}

display(head);

system("pause");

return 0;

}

**Результаты программы**

