**Задание № 2**

**Индивидуальное задание:**

Написать программу по созданию, добавлению, просмотру и решению поставленной задачи (в рассмотренных примерах это действие отсутствует) для однонаправленного линейного списка типа СТЕК. Реализовать сортировку стека двумя рассмотренными выше методами.

Решение поставленной задачи представить в виде блок-схемы.

Во всех заданиях создать список из положительных и отрицательных случайных целых чисел.

* 1. **Создание консольного приложения**

Код программы:

#include <stdio.h>

#include <stdbool.h>

#include <locale.h>

#define STACK\_OVERFLOW -100

#define STACK\_UNDERFLOW -101

#define OUT\_OF\_MEMORY -102

typedef int T;

typedef struct Node\_tag

{

T info;

struct Node\_tag\* next;

} Node\_t;

void push(Node\_t\*\* head, T info);

Node\_t\* pop1(Node\_t\*\* head);

void SortStackByAdress(Node\_t\* srtStack);

void (\*SortStackByAdres) (Node\_t\* srtStack);

void SortStackByData(Node\_t\* srtStack);

void printStack(const Node\_t\* head);

size\_t getSize(const Node\_t\* head);

Node\_t\* swapMinAndMax(Node\_t\* Head);

int main()

{

setlocale(LC\_ALL, "Russian");

Node\_t\* head = NULL;

char command = '\\';

int info;

puts(" 1 - Добавить в стек\n 2 - Достать из стека\n 3 - Вывести стек\n 4 - Поменять местами максимальный и минимальный элементы\n 5 - Сортировка по адресу\n 6 - Сортировка по значению\n 7 - Выход\n");

while (command != '7')

{

scanf\_s("%1c", &command);

switch (command)

{

case '1':

puts("Введите значение");

scanf\_s("%4d", &info);

push(&head, info);

break;

case '2':

printf("%d\n", pop1(&head)->info);

break;

case '3':

printStack(head);

break;

case '4':

head = swapMinAndMax(head);

printStack(head);

break;

case '5':

SortStackByAdres(head);

break;

case '6':

SortStackByData(head);

break;

default:

break;

}

}

getch();

return 0;

}

void push(Node\_t\*\* head, T info)

{

Node\_t\* tmp = malloc(sizeof(Node\_t));

if (tmp == NULL)

{

exit(STACK\_OVERFLOW);

}

SortStackByAdres = SortStackByData;

tmp->next = \*head;

tmp->info = info;

\*head = tmp;

}

Node\_t\* pop1(Node\_t\*\* head)

{

Node\_t\* out;

if ((\*head) == NULL)

{

exit(STACK\_UNDERFLOW);

}

out = \*head;

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

return out;

}

void SortByAdress(Node\_t\* srtStack)

{

Node\_t\* t, \* m, \* a, \* b;

for (bool go = true; go; )

{

go = false;

a = t = srtStack->next;

b = srtStack->next->next;

while (b != NULL)

{

if (a->info > b->info)

{

if (t == a)

srtStack->next = b;

else

t->next = b;

a->next = b->next;

b->next = a;

m = a, a = b, b = m;

go = true;

}

t = a;

a = a->next;

b = b->next;

}

}

}

void SortStackByData(Node\_t\* srtStack)

{

Node\_t\* a;

T tmpData;

for (bool go = true; go; )

{

go = false;

a = srtStack;

while (a->next != NULL)

{

if (a->info > a->next->info)

{

tmpData = a->info;

a->info = a->next->info;

a->next->info = tmpData;

go = true;

}

a = a->next;

}

}

}

void printStack(const Node\_t\* head)

{

printf("stack > ");

while (head) {

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

head = head->next;

}

printf("\n");

}

Node\_t\* swapMinAndMax(Node\_t\* Head)

{

int MAX;

int MIN;

Node\_t\* ptr = Head;

MIN = ptr->info;

MAX = ptr->info;

while (ptr) {

if (ptr->info > MAX) {

MAX = ptr->info;

}

if (ptr->info < MIN) {

MIN = ptr->info;

}

ptr = ptr->next;

}

printf\_s("Максимальное число стека: %d\n Минимальное число стека: %d\n", MAX, MIN);

ptr = Head;

while (ptr) {

if (ptr->info == MAX) {

ptr->info = MIN;

}

else if (ptr->info == MIN) {

ptr->info = MAX;

}

ptr = ptr->next;

}

return Head;

}

size\_t getSize(const Node\_t\* head)

{

size\_t size = 0;

while (head)

{

size++;

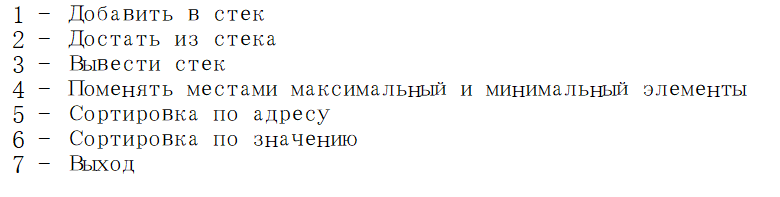
head = head->next;

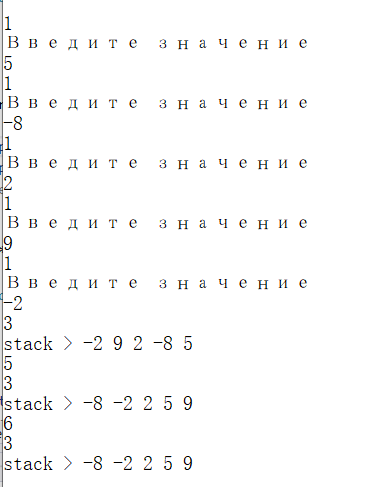
}

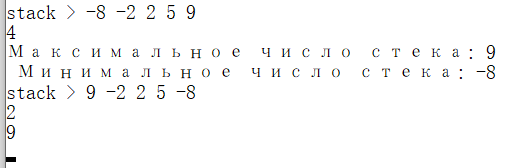
return size;

}

**Результаты выполнения:**







**Блок-схема:**

