**КР № 2**

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**

Текст программы:

#include <stdio.h>

#include <stdbool.h>

#define STACK\_OVERFLOW -100

#define STACK\_UNDERFLOW -101

#define OUT\_OF\_MEMORY -102

typedef int T;

typedef struct Node\_tag

{

T value;

struct Node\_tag\* next;

} Node\_t;

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

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

void SortStackByAdress(Node\_t\* srtStack);

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

void SortStackByData(Node\_t\* srtStack);

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

{

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

if (tmp == NULL)

{

exit(STACK\_OVERFLOW);

}

SortStackByAdres = SortStackByData;

tmp->next = \*head;

tmp->value = value;

\*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 SortStackByAdress(Node\_t\* srtStack)

{

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

for (bool go = true; go; )

{

go = false;

a = t = srtStack;

b = srtStack->next;

while (b != NULL)

{

if (a->value > b->value)

{

if (t == a)

srtStack = 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->value > a->next->value)

{

tmpData = a->value;

a->value = a->next->value;

a->next->value = tmpData;

go = true;

}

a = a->next;

}

}

}

void printStack(const Node\_t\* head)

{

printf("stack > ");

while (head) {

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

head = head->next;

}

printf("\n");

}

Node\_t\* dltNegativeElems(Node\_t\* head)

{

Node\_t\* temp = head;

Node\_t\* prev = temp;

while (temp)

{

if (temp->value < 0)

{

if (temp == head)

{

head = head->next;

Node\_t\* del = temp;

temp = temp->next;

free(del);

continue;

}

else

{

prev->next = temp->next;

Node\_t\* del = temp;

temp = prev;

free(del);

}

}

prev = temp;

temp = temp->next;

}

return head;

}

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

{

size\_t size = 0;

while (head)

{

size++;

head = head->next;

}

return size;

}

int main()

{

Node\_t\* head = NULL;

char command = '\\';

int value;

puts(" 1 - Push\n 2 - Pop\n 3 - Print\n 4 - Delete negative elements\n 5 - Sort by adress\n 6 - Sort by value\n 7 - Exit\n");

while (command != '7')

{

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

switch (command)

{

case '1':

puts("Input value");

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

push(&head, value);

break;

case '2':

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

break;

case '3':

printStack(head);

break;

case '4':

head = dltNegativeElems(head);

printStack(head);

break;

case '5':

SortStackByAdres(head);

break;

case '6':

SortStackByData(head);

break;

default:

break;

}

}

puts("Press any key to continue...");

getch();

return 0;

}

Результаты выполнения:







