**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Лабораторная работа №4

по дисциплине «Программирование»

Указатели

Группа: АВТ-041

Студенты: Антонов А.

Преподаватель: Балакин В.

НОВОСИБИРСК 2021

**Задача**

Сортировка двусвязного списка выбором.

**Проектирование программы**

Идея: Создание двусвязного списка с двумя сторожами head и tail

Составные части программы:

1. Добавление элемента в двусвязный список и возвращение его адреса

List\* InsertElementInList(List\* last, List\* p)

{

if (last && p)

{

p->pred = last;

p->next = last->next;

last->next = p;

p->next->pred = p;

return p;

}

else

return NULL;

}

1. Функция создания двусвязного списка, по входной последовательности целых чисел, вводимых с консоли до нуля:

void CreateList(List\*& head, List\*& tail)

{

head = new List;

tail = new List;

head->next = tail;

tail->pred = head;

int k;

printf("Вводите целые числа до нуля ");

scanf\_s("%d", &k);

List\* last = head;

while (k)

{

List\* p = new List;

p->info = k;

last = InsertElementInList(last, p);

scanf\_s("%d", &k);

}

return;

}

1. Вывод двусвязного списка на консоль:

void PrintList(List\* head, List\* tail)

{

List\* p = head->next;

while (p != tail)

{

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

p = p->next;

}

printf("\n");

return;

}

1. Функция сортировки элементов. Учитывается положение указателей с предыдущих на новые элементы:

void SortList(List\*& head, List\*& tail) {

//Код аналогичен созданию списка. Создаём новый список,

//в который уже в нужном порядке будем добавлять элементы

List\* head2 = new List;

List\* tail2 = new List;

List\* last = head2;

head2->next = tail2;

tail2->pred = head2;

while (head->next != tail)//пока в исходном списке остались элементы

{ //ищем минимальный из оставшихся и удаляем

List\* min = head->next, \* p = head->next;

while (p != tail)

{

if (p->info < min->info)

min = p;

p = p->next;

}

//разрыв связей, предыдущий и следующий элемент указывают друг на друга в обход min

min->next->pred = min->pred;

min->pred->next = min->next;

last = InsertElementInList(last, min);//найденный элемент добавляем в начало нового списка

}

head = head2;

tail = tail2;

}

1. Функция main:

int main()

{

setlocale(LC\_ALL, "Russian");

List\* head, \* tail = NULL;

CreateList(head, tail);

printf("Исходный список ");

PrintList(head, tail);

///////////////

SortList(head, tail);

printf("Сортированный список ");

PrintList(head, tail);

system("pause");

}

Переменные:

head, tail - сторожи

k - кол-во чисел

pred, next - указатели на предыдущий и следующий элементы

**Текст программы с комментариями**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct List {

int info;

List\* pred, \* next;

};

// Функция вставляет элемент p в двусвзный список посел элмента last

// и возвращает адрес вставленного элемента

List\* InsertElementInList(List\* last, List\* p)

{

if (last && p)

{

p->pred = last;

p->next = last->next;

last->next = p;

p->next->pred = p;

return p;

}

else

return NULL;

}

//Функция создает двусвязный список с двумя сторожами head and tail

// по входной поледовательности целых чисел, вводимх с консоли до нуля

void CreateList(List\*& head, List\*& tail)

{

head = new List;

tail = new List;

head->next = tail;

tail->pred = head;

int k;

printf("Вводите целые числа до нуля ");

scanf\_s("%d", &k);

List\* last = head;

while (k)

{

List\* p = new List;

p->info = k;

last = InsertElementInList(last, p);

scanf\_s("%d", &k);

}

return;

}

// Фунция выводит на консоь двусвзяный список со сторожами head and tail

void PrintList(List\* head, List\* tail)

{

List\* p = head->next;

while (p != tail)

{

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

p = p->next;

}

printf("\n");

return;

}

void SortList(List\*& head, List\*& tail) {

//Код аналогичен созданию списка. Создаём новый список,

//в который уже в нужном порядке будем добавлять элементы

List\* head2 = new List;

List\* tail2 = new List;

List\* last = head2;

head2->next = tail2;

tail2->pred = head2;

while (head->next != tail)//пока в исходном списке остались элементы

{ //ищем минимальный из оставшихся и удаляем

List\* min = head->next, \* p = head->next;

while (p != tail)

{

if (p->info < min->info)

min = p;

p = p->next;

}

//разрыв связей, предыдущий и следующий элемент указывают друг на друга в обход min

min->next->pred = min->pred;

min->pred->next = min->next;

last = InsertElementInList(last, min);//найденный элемент добавляем в начало нового списка

}

head = head2;

tail = tail2;

}

int main()

{

setlocale(LC\_ALL, "Russian");

List\* head, \* tail = NULL;

CreateList(head, tail);

printf("Исходный список ");

PrintList(head, tail);

///////////////

SortList(head, tail);

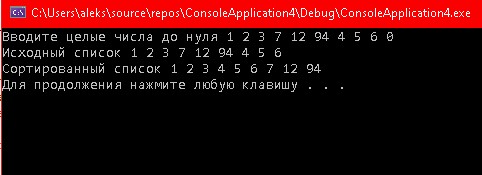
printf("Сортированный список ");

PrintList(head, tail);

system("pause");

}

**Пример работы программы**



**Вывод**  
Функция была реализована, программа работает корректно.