Лабораторная №12. Структуры в динамической памяти. Линейные списки.

Темы, рассматриваемые в лабораторной работе:

Организация динамической структуры данных с использованием указателя. Основные алгоритмы работы с линейными односвязными списками. Контроль корректности работы с динамической памятью.

Во всех задачах предполагается использование линейного односвязного списка. Информационное поле элемента списка может быть целым числом, символом или строкой string. Например,

```
// Узел списка:
//     data - значение в узле списка;
//     next - указатель на следующий элемент.
struct elem_list
{
    typedef int datatype; //     datatype - тип данных списка;
    datatype data;
    elem_list *next;
};
```

При написании тестов можно создать вспомогательные функции: создающую список из элементов массива и перемещающую элементы списка в массив.

Задания:

1) Создать проект из заголовочного и исполнимого файлов в архиве Lab12.zip.

    (1) Подготовить проект к тестированию работы с динамической памятью. Для этого в исходном файле source.cpp поставить точки останова в строках с номерами 10, 19, 46, 53, 60, 62.
    (2) Запустить проект в режиме Debug, на первой точке останова настроить режим управления снимками памяти, сделать первый снимок и поместить в окно контрольных значений переменные l1, p, l2, t.
    (3) Перемещаясь от одной точки останова к другой через команду «продолжить», получать в каждой точке останова снимки памяти и следить за изменением состоянии переменных в окне просмотра контрольных значений.
    (4) При повторном запуске отладчика выполнить пошаговую трассировку одного из циклов создания списка и удаления списка (для упрощения можно уменьшить количество элементов в списке).

Остальные задания выполняются уже без контроля утечек памяти, поэтому можно изменить описание структуры elem_list, убрав из нее поле buf.

    (5) Раскомментировать фрагмент программы, начиная со строки 30 до строки 44. Убрать точки останова и выполнить всю программу целиком.
    (6) Изменить в примере создание списка таким образом, чтобы можно было вводить длину списка и элементы списка с консоли.

При написании функций можно использовать как список целых чисел, так и список, элементы которого имеют тип datatype. Для функций предусмотреть тесты.

2) Функция создания списка заданной длины, элементы списка вводятся с консоли. Порядок следования элементов должен совпадать с порядком ввода.

```
// Данные вводятся с консоли.
// Входные параметры:
//     length - количество элементов списка.
// Возвращает указатель на первый элемент списка.
elem_list  *read_list(int length);
```

3) Функция удаления списка.

```
// Входные параметры:
//     begin - указатель на начало списка.
// Выходное значение параметра begin равно nullptr.
void delete_list(elem_list  *&begin);
```

4) Функция вывода списка на консоль.

```
// Входные параметры:
//     begin - указатель на начало списка.
void print(const elem_list  *begin);
```

5) Количество элементов списка

```
// Входные параметры:
//     begin - указатель на начало списка
// Возвращает количество элементов списка
int length(const elem_list  *begin);
```

6) Функция поиска элемента

```
// Входные параметры:
//     begin - указатель на начало списка;
//     x - искомое значение.
// Возвращает указатель на первый элемент с заданным значением или
// nullptr, если элемента с таким значение в списке нет.
elem_list  *find(const elem_list  *begin, datatype x);
```

7) Значение первого элемента списка

```
// Входные параметры:
//     begin - указатель на начало списка
// Возвращает значение первого элемента списка
// или исключение, если список пуст
datatype first(const elem_list  *begin);
```

8) Вставка элемента в начало списка.

```
// Входные параметры:
//     begin - указатель на начало списка;
//     x - новое значение.
// Возвращает указатель на новое начало списка.
elem_list  *add_first(elem_list  *&begin, datatype x);
```

9) Вставка элемента в конец списка.

```
// Входные параметры:
//     begin - указатель на начало списка;
//     x - новое значение.
// Возвращает указатель на новый элемент списка.
elem_list  *add_last(elem_list  *&begin, datatype x);
```

10) Вставка нового значения перед указанным элементом списка.

```
// Входные параметры:
//     begin - указатель на начало списка;
//     item - указатель на элемент для вставки;
//     x - новое значение.
// Возвращает указатель на новый элемент списка.
elem_list  *insert_before(elem_list  *&begin, elem_list  *item, datatype x);
```

11) Удаление элемента после указанного элемента списка.

```
// Входные параметры:
//     item - указатель на элемент для удаления.
void remove_after(elem_list  *item);
```

Дополнительные задания:

12) Вставить новый элемент в список за первым элементом с заданным значением. Если вставить невозможно, функция должна выбрасывать исключение.
13) Удалить из списка все элементы с заданным значением.
14) Поверить, что список целых чисел упорядочен по возрастанию значений его элементов.
15) Проверить список на выполнение условия для элементов списка, условие задается функцией –параметром.
16) «Перевернуть» список, не используя операций выделения/освобождения динамической памяти.