Занятие 5. Однонаправленный динамический список

Цель: получить знания и практические навыки управления динамическим однонаправленным списком.

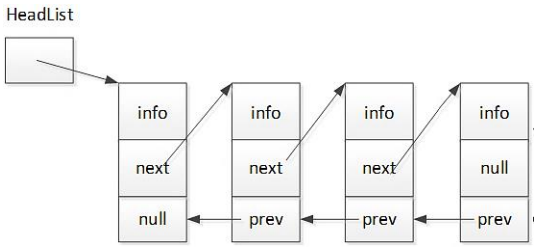
1 Однонаправленный список средствами языка C++

Связные линейные списки предназначены для хранения многоэлементных последовательностей. Ячейка (элемент списка, узел) хранит значение (информационная часть) и связи (ссылки) с другими ячейками списка.

Не существует понятия позиции (индекса) узла. Для организации таких структур в ЯВУ C++ используется механизм указателей.

В зависимости от количества связей в элементе различают: однонаправленные (или односвязные) и двунаправленные (двухсвязные) списки (соответственно, рис. 1.а и 1.б).

а) Односвязный список



б) Двусвязный список

Рисунок 1. Линейные динамические списки

Линейные структуры данных: списки, стеки (метод LIFO), очереди (метод FIFO), деки.

Способы представления структур: пользовательские типы (массивы и классы на основе указателей).

Базовые операции над структурами данных:

- найти нужный элемент в структуре;
- обратиться к нужному элементу структуры;
- добавить элемент в структуру;
- удалить элемент из структуры.

В C++ имеется удобный и мощный механизм реализации динамических структур – указатели и структурный тип (или абстрактные типы данных).

Начать лучше с описания типа данных для узла списка (листинг 1).

Листинг 1. Реализация узла однонаправленного списка

```
struct Node {
  string val;
  Node* next;
  Node(string _val) : val(_val), next(nullptr){}
};
```

Здесь поле string – информационное (полезная строка данных), next – связь, Node – конструктор экземпляра узла списка.

Далее реализуем сам односвязный список. В списке будет реализовано:

1) указатель на первый узел;
2) указатель на последний узел;
3) конструктор списка;
4) функция проверки наличия узлов в списке;
5) функция добавления элемента в конец списка;
6) функция печати списка;
7) функция поиска узла в списке по ключу;
8) функция удаления узла по ключу.

В следующем коде реализуем пункты 1-3 (листинг 2).

Листинг 2. Описание типа данных для односвязного списка

```
struct list {
    Node* first;
    Node* last;
    list() : first(nullptr), last(nullptr) {}
};
```

В код главной функции main необходимо добавить создание экземпляра списка, например:

list l;

Добавим в структуру list вспомогательную функцию для проверки наличия узлов в списке – однострочная, в ней необходимо проверить не пуст ли первый узел (листинг 3).

Листинг 3. Проверка наличия узлов в односвязном списке

```
bool is_empty() {
    return first == nullptr;
}
```

Реализуем в структуре list функцию добавления элемента в конец списка. Здесь возможны два случая: а) список пустой и б) список не пустой.

В обоих случаях необходимо создать сам узел с заданным (переданным в функцию) значением.

Для первого случая понадобится определённая ранее функция проверки наличия узлов. Если список пуст, указатели на первый и последний узел направляем на единственный новый узел.

Для второго случая нужно указать, что новый узел стоит после последнего узла, после чего меняем значения указателя на последний узел last (листинг 4).

Листинг 4. Добавление узла в конец списка.

```
void push_back(string _val) {
    Node* p = new Node(_val);
    if (is_empty()) {
      first = p;
      last = p;
      return;
    }
    last->next = p;
    last = p;
}
```

Т.о. в список можно добавлять и другие узлы.

В структуре list в функции печати всего списка (если список не пуст) направляем указатель p на первый узел списка и выводим значения узлов, пока указатель p не пустой. При каждой итерации перенаправляем p на следующий узел (листинг 5).

Листинг 5. Вывод списка в консоль.

```
void print() {
    if (is_empty()) return;
    Node* p = first;
    while (p) {
        cout << p->val << " ";
        p = p->next;
    }
    cout << endl;
}
```

Для поиска узла в списке по ключевому значению в структуре list добавим функцию обхода списка, пока указатель p не пустой и пока значение узла p не равно ключу, после чего возвращаем найденный узел, если он есть (лист. 6).

Листинг 6. Поиск узла в списке по ключу.

```
Node* find(string _val) {
    Node* p = first;
    while (p && p->val != _val) p = p->next;
        return (p && p->val == _val) ? p : nullptr;
}
```

Листинг 7. Удаление узла из списка по ключу.

```
void remove(string _val) {
    if (is_empty()) return;
    if (first->val == _val) {
        remove_first();
        return;
    }
    else if (last->val == _val) {
            remove_last();
        return;
    }
    Node* slow = first;
    Node* fast = first->next;
    while (fast && fast->val != _val) {
        fast = fast->next;
        slow = slow->next;
    }
    if (!fast) {
        cout << "This element does not exist" << endl;
        return;
    }
    slow->next = fast->next;
    delete fast;
}
```

Добавим в структуру list функцию удаления узла из списка по заданному ключу.

Если список не пуст, то возможны три случая:

1) узел с искомым значением равен первому;
2) узел с искомым значением равен последнему;
3) не первый и не второй случаи.

Первый случай: сравниваем значение первого узла с ключом, если значения совпадают, тогда вызываем функцию удаления первого узла.

Второй случай: сравниваем значение последнего узла с ключом, если значения совпадают, тогда вызываем функцию удаления последнего узла.

Третий случай:

Создаются указатели slow на первый узел, и fast – на следующий после первого. Затем, пока fast не пустой и пока значение текущего узла fast не равно ключу, при каждой итерации перенаправляем slow и fast на следующий после них узел. Если указатель fast пустой, то сообщаем об ошибке, иначе просто удаляем узел fast.

Описанный код вместе с вспомогательными функциями удаления первого и последнего узлов в списке, приведён в листинге 7.

Здесь приведён один из возможных способов реализации базовых операций с односвязным списком. Возможны и другие реализации, в т.ч на основе абстрактных типов данных в рамках ООП.

Базовые операции с другими типами линейных структур (двунаправленный список, стек, очередь, дек) реализуются аналогичным образом с поправкой на их специфику.

2 Задание

Реализуйте программу решения задачи варианта по использованию линейного однонаправленного списка.

Требования для всех вариантов

1. Информационная часть узла определена вариантом
2. Разработать функции вставки нового узла перед первым узлом и удаления узла по ключу.
3. Реализуйте возможность а) создания нового списка вручную, а также б) использования уже готового списка для тестирования заданий индивидуального варианта.
4. Разработать функцию вывода списка в консоль.
5. Разработать функции согласно индивидуальному варианту. При необходимости можно добавлять вспомогательные функции, декомпозируя задачу.
6. Реализуйте текстовое пользовательское меню.
7. В основной программе выполните тестирование каждой функции (пункты 2-5).
8. Составить отчет по выполненному заданию (формат отчета после вариантов).

3 Варианты

№ вар.	Тип информационной части узла	Дополнительные операции
1	int	Даны два линейных однонаправленных списка L1 и L2. 1. Разработать функцию, которая формирует список L, включив в него по одному разу элементы, значения которых входят хотя бы в один из списков L1 и L2. 2. Разработать функцию, которая удаляет из списка L1 все узлы в четных позициях. 3. Разработать функцию, которая вставляет в список L2 после каждой пары узлов новый узел со значением равным сумме значений двух предыдущих узлов. Если количество узлов в исходном списке нечетное, то после последнего узла новый узел не вставлять
2	float	Даны два линейных однонаправленных списка L1 и L2. 1. Разработать функцию, которая формирует список L, включив

		в него по одному разу элементы, значения которых входят одновременно в оба списка L1 и L2. 2. Разработать функцию, которая удаляет узел списка L2, расположенный перед узлом, содержащим отрицательное значение. И так для всех узлов, содержащих отрицательное значение. 3. Разработать функцию, которая вставляет новый узел с заданным значением перед каждым узлом списка L1, содержащим нечетное значение.
3	char	Даны два линейных однонаправленных списка L1 и L2. 1. Разработать функцию, которая формирует список L, включив в него по одному разу элементы, значения которых входят в список L1 и не входят в список L2. 2. Разработать функцию, которая удаляет подсписок списка L1 заданный диапазоном позиций. Например, со второго три. 3. Разработать функцию, которая упорядочивает значения списка L2, располагая их в порядке возрастания.
4	int	Дан линейный однонаправленный список L1 1. Разработать функцию, которая переформирует список L1, переписав в начало списка его часть, начиная с заданной позиции (номер позиции передается в функцию). 2. Разработать функцию вставки узла в упорядоченный по не возрастанию список. Сформировать такой список L2. 3. Разработать функцию, которая удаляет из L2 все повторяющиеся значения, оставляя одно из них.
5	char	Даны два линейных однонаправленных списка L1 и L2 с головным элементом. 1. Разработать функцию, которая проверяет на равенство списки L1 и L2. 2. Разработать функцию, которая вставляет в список L1 последний элемент списка L2. 3. Разработать функцию, которая удаляет из списка L2, узлы, содержащие цифровые значения.
6	double	Дан линейный однонаправленный список L 1. Разработать функцию, которая вставляет перед последним узлом два новых узла. 2. Удаляет из списка L первое отрицательное значение, если оно присутствует в списке. 3. Найти в списке L максимальное значение и перенести его узел в конец списка.
7	int	Дан линейный однонаправленный список L 1. Разработать функцию, которая проверяет, есть ли в списке L два одинаковых элемента.

		2. Разработать функцию, которая удаляет из списка L максимальное значение. 3. Разработать функцию, которая вставляет в список L новое значение перед каждым узлом в четной позиции.
8	float	Дан линейный однонаправленный список L 1. Разработать функцию, которая переносит первые k узлов в конец списка. 2. Разработать функцию, которая переставляет местами узлы с максимальным и минимальным значениями. 3. Разработать функцию, которая удаляет предпоследний узел списка.
9	char	Дан линейный однонаправленный список L, содержащий текст. В каждом узле один символ. Слова разделены одним пробелом. 1. Разработать функцию, которая находит последнее слово и переставляет его в начало списка. 2. Разработать функцию, которая удаляет второе слово. 3. Разработать функцию, которая заменяет k-ое слово на новое слово. Длина нового слова может быть больше длины k-ого слова.
10	char	Дан линейный однонаправленный список L 1. Разработать функцию, определяющую в списке L самую длинную последовательность одинаковых символов. 2. Разработать функцию, которая в каждой последовательности одинаковых символов оставляет только один. 3. Разработать функцию, которая создает новый список из цифр исходного, выполняя вставку элемента в новый список в порядке возрастания цифр. В новом списке не может быть повторяющихся цифр.
11	int	Дан линейный однонаправленный список L, информационная часть которого содержит однозначные и двузначные числа. 1. Разработать функцию, которая создает массив A из 10 указателей на элемент списка и включает в список элемента массива с индексом i, числа списка L, которые начинаются с цифры равной i. Включение в конец списка. Однозначные числа включаются в список массива с индексом 0. 2. Разработать функцию, которая удаляет список L. 3. Разработать функцию, которая создает список L, включая в него списки массива A последовательно от списка с индексом 0 до списка с индексом 9.
12	int	Дан линейный однонаправленный список L1, информационная часть которого содержит однозначные и двузначные числа, упорядоченные в порядке возрастания старшей цифры.

		1. Разработать функцию, которая удаляет узел в заданной позиции списка L1. 2. Разработать функцию, которая формирует новый список L2 вставляя в него элементы списка L1, располагая их в порядке возрастания младшей цифры. Удаляя из списка L1 перемещенный узел. 3. Разработать функцию, которая определяет, что список L2 упорядочен по возрастанию.
13	int	Дан массив из n указателей на вершины списков. Структура узла списка содержит ключ (информационная часть узла) и ссылку на следующий узел. 1. Разработать функцию, которая вставляет переданный в качестве параметра ключ в i-ый список массива. Индекс i определяется по правилу: i=key%n. Некоторые элементы массива могут остаться nullptr. 2. Разработать функцию для удаления значение ключа из списка. 3. Разработать функцию, которая находит узел со значением ключа и возвращает указатель на найденный узел.
14	int	Дан линейный однонаправленный список L 1. Разработать функцию, которая создает из значений узлов списка L два новых списка: L1 – из положительных элементов массива L; L2 – из остальных элементов списка L. 2. Разработать функцию, которая удаляет из списка L2 все отрицательные элементы. 3. Разработать функцию, которая в списке L1 узел с максимальным значением размещает перед первым узлом.
15	int	Дан линейный однонаправленный список L, узлы которого упорядочены по возрастанию в соответствии со значениями информационной части узла. 1. Разработать функцию, которая вставляет новое значение в список L, сохраняя упорядоченность списка. 2. Разработать функцию, которая удаляет из списка L все узлы, значения в которых большие заданного. 3. Разработать функцию, которая создает новый список L1 из значений узлов списка L, так что в списке L2 узлы упорядочены в порядке убывания их значений.
16	char	Даны два линейных однонаправленных списка L1 и L2. 1. Разработать функцию, которая вставляет в список L1 за узлом с заданным значением X все узлы списка L2, если узел X есть в списке L1. 2. Разработать функцию, которая из списка L2, удаляет все узлы со значением, не являющимся цифрой.

		3. Разработать функцию, которая из цифр списка L2 образует целое число допустимой разрядности.
17	<степень, коэффициент> <int, real>	Линейный многочлен n-ой степени представлен в программе как линейный однонаправленный список. Каждый i-ый узел списка хранит информацию по i-му члену многочлена. Поэтому информационная часть узла состоит из двух значений: степень члена и коэффициент при этой степени. Если i-ый член в многочлене отсутствует, то узел не создается. 1. Разработать функцию, которая создает список по переданному в качестве параметра многочлену: он представлен массивом коэффициентов и их степеней. 2. Разработать функцию, которая выводит многочлен и представляет его в форме выражения. 3. Разработать функцию, которая вычисляет значение многочлена при заданном значении x. В вычислении использовать алгоритм Горнера.

4 Форма отчета по заданию

Условие задания, требования в соответствии с вариантом

1. Постановка задачи
2. Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания.
   2.1 Определить структуру узла однонаправленного списка в соответствии с вариантом.
   2.2 Изобразить (рисунок) для каждой операции полученного списка процесс выполнения операции на существующем однонаправленном списке.
   2.3 Изобразите структуру данных, которая будет использоваться в операциях.
   2.4 Привести алгоритм выполнения операции
   2.5 Привести таблицу тестов для тестирования каждой операции
3. Представить код программы
4. Представить результат тестирования программы: скриншоты выполнения каждой операции.
5. Привести выводы по полученным знания и умениям
6. Список информационных источников, которые были использованы при выполнении задания.

5 Контрольные вопросы

1) Расскажите о трех уровнях представления данных в программной системе.
2) Что определяет тип данных?
3) Что определяет структура данных?
4) Расскажите о структуры хранения данных в компьютерных технологиях.
5) Дайте определение линейной структуре данных.
6) Дайте определение структуре данных линейный список.
7) Дайте определение структуре данных стек.
8) Дайте определение структуре данных очередь.
9) Чем стек отличается от структуры данных линейный список?
10) Какой из видов линейных списков лучше использовать, если нужно введенную последовательность вывести наоборот?
11) Определите сложность алгоритма операции вставки элемента в i-ую позицию: а) массива; б) линейного списка.
12) Определите сложность алгоритма операции удаления элемента из i-ой позиции: а) массива; б) линейного списка.
13) В чем суть трюка Вирта при выполнении операции удаления элемента из списка?
14) Определите структур узла однонаправленного списка.
15) Реализуйте алгоритм вывода линейного однонаправленного списка.
16) Приведите фрагмент кода программы на языке C++ выполнения операции перемещения последнего элемента в начало списка.
17) Какое из действий лишнее в следующем фрагменте кода? Куда вставляется новый узел?

```
struct Node{
        int info;
  Node*next;
};
typedef Node *List;
List L=new List;
void insertToList(List LL, int x){
    List q=new Node; q->info=x; q->next=0;
    if (LL==nullptr) LL->next=q;
    else
      q->next=LL->next;
            LL->next=q;
}
```