Практическое занятие №1

Подготовка к 1-ой лабораторной работе.

Реализация основных алгоритмов.

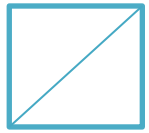
Основные термины

1. Информация в таблице представлена множеством **узлов** (записей, объектов, элементов).

Узел обозначается так: 

2. Каждый узел состоит из одного или нескольких последовательных слов в памяти машины, разделенных на именуемые части, называемые **полями**.

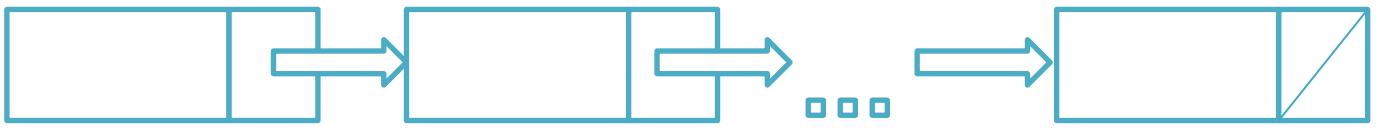
3. Адресом узла является адрес первого слова в узле (связь, указатель, ссылка, стрелка на узел).

Для обозначения пустой связи используется: 

Определение линейного списка

Линейный список – это множество, состоящее из $n \geq 0$ узлов: x[0], …, x[n-1], структурные свойства которого ограничиваются условиями:

1. если $n \geq 0$, то x[0] является первым
2. если $0 < k < n-1$, то k-му узлу предшествует x[k-1] и за ним следует x[k+1].



Организация хранения данных

Для хранения данных целесообразно использовать следующую конструкцию:

```
struct list  {
    int inf; //информационная часть
    list * next; //ссылка на следующий элемент
};
list * first;
```

Генерация псевдослучайных чисел

1. Используется команда rand()
2. Заголовочный файл <stdlib.h>
3. Для того, чтобы генератор каждый раз начинал генерацию с нового числа, необходимо подключить заголовочный файл <time.h>
4. Для получения различных чисел:
   srand((unsigned)time(0));
5. Для генерации целого числа от a до b
   int irand(int a, int b)
   { return rand() % (b-a+1)+a; }

Пример: *получить псевдослучайное число от -7 до 4*

irand(-7, 4)

Создание линейного списка

```
n=7;
list *first=new list;
list *p=first;
cin>>p->inf ;
for (int i=0; i<n-1; i++)
{
  p->next=new list ;
  p=p->next ;
  cin>>p->inf;
}
p->next=0;
```

Здесь описано построение линейного списка из n (n=7) элементов.

Элементы списка вводятся с клавиатуры.

cin - заставляет программу ожидать ввода числа от пользователя.

Cin – *объект, определенный в С++ для работы со стандартным потоком ввода.*

>> - *операция извлечения.*

Приведенный код удобно использовать, когда число элементов в списке известно заранее.

Подумайте, как нужно модифицировать программу, чтобы она работала для произвольного числа элементов.

Печать линейного списка

```
p = first;
while (p)
{
  cout << p->inf << “ “ ;
  p = p->next;
}
cout << endl;
```

Данный код работает для произвольного числа элементов в списке.

Идентификатор cout *является объектом C++, предназначенным для работы со стандартным потоком вывода.*

Оператор ***<<*** *называется операцией вставки.*

Манипулятор endl *– вставка в символьный поток символа окончания строки.*

Манипулятор *- это особая инструкция, обращенная к потоку и предназначенная для изменения вывода.*

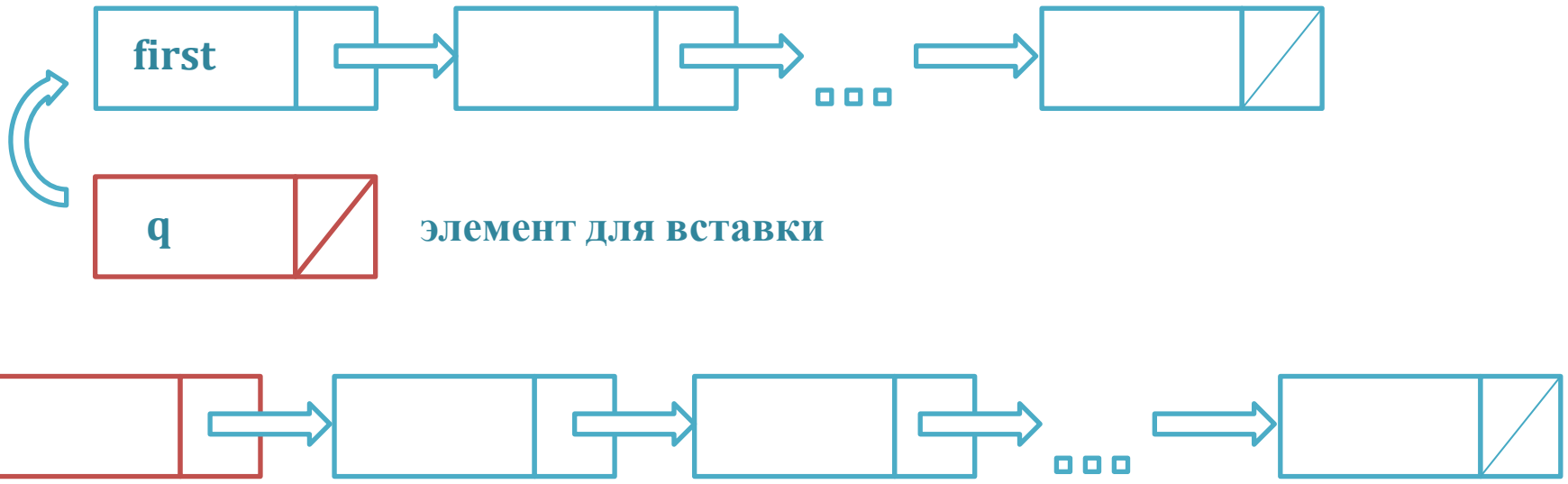
Создание пользовательского интерфейса

```
int num = 1;
while (num) {
  cin >> num;
  switch (num) {
    case 0: break;
    case 1: {
              //ввод списка
              break; }
    case 2: {
              //вывод списка
              break; }
    default: cout << “Error!” << end l;
    }
}
```

Вставка элемента в начало списка

```
list * q = new list;
cin >> q->inf;
q->next = first;
first = q;
```

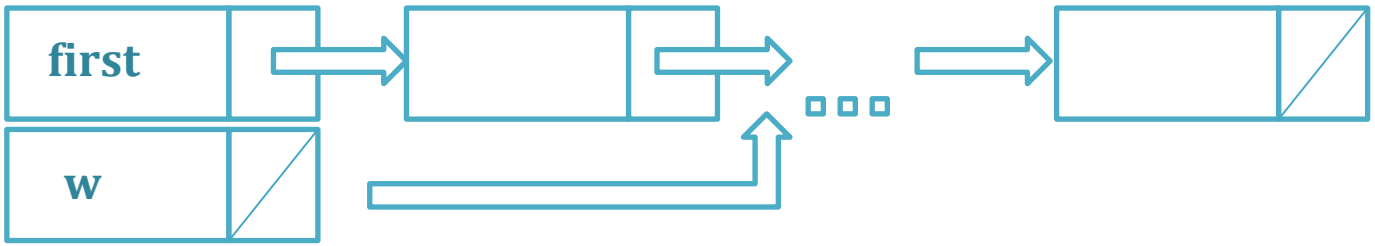
Рассмотрен случай, когда список не пуст.



Включение элемента в середину (после i-го элемента)

```
p = first;
int k = 2;
for (i=0; i<k; i++)
   p = p->next;
list *w=new list;
cin >> w->inf ;
w->next = p->next;
p->next = w;
```

Рассмотрен случай, когда список не пуст.

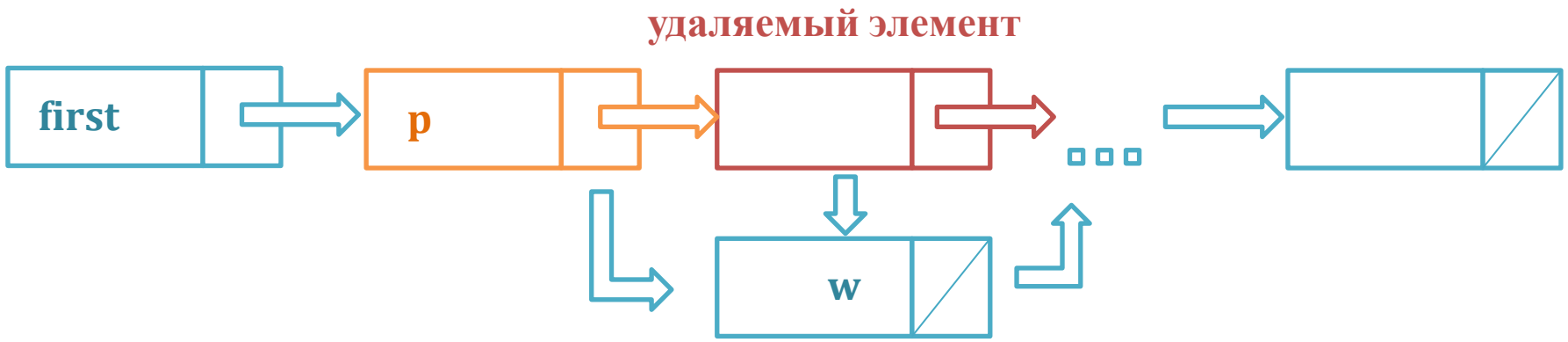


Удаление элемента из середины (i-го элемента)

```
k = 2;
p = first;
for (i=0; i<k-1; i++ )
   p = p->next;
w = p->next;
p->next = w->next;
delete w;
```

Используется отдельная переменная для сохранения удаляемого элемента.

Это необходимо для того, чтобы память не оставалась помеченной как занята программой.



Создание копии списка

```
p = first;
list *newlist =new list;
newlist->inf = p->inf;
p = p->next;
list *newfirst = newlist;
while (p) {
  newlist->next = newlist;
  newlist = newlist->next;
  newlist->inf = p->inf;
  p = p->next ;
}
newlist->nex t = 0;
```

Создание копии списка происходит фактически (а не только созданием нового начала и присоединения к нему остатка списка).

Код работает, только если список содержит хотя бы 1 элемент.

Разбиение списка на два по заданному критерию

```
p = first;
list *chet = newlist;
list *nechet = newlist;
list *p1 = chet;
list *p2 = nechet;
list *w1 = p1;
list *w2 = p2;
while (p) {
if (p->inf % 2) {
   p2->inf = p->inf;
   p2->next = newlist;
   w2 = p2;
p2 = p2->next;
}
```