# Односвязный список

Список - это динамическая структура данных, т.е.

* Память выделяется (освобождается) во время выполнения
* Размер списка может быть изменен (увеличен/уменьшен)

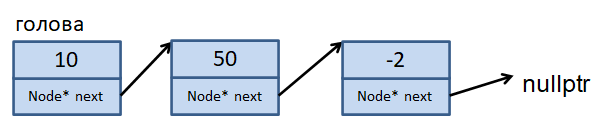
Список состоит из элементов одного типа, связанных между собой последовательно с помощью указателей.

Элементы списка называются «узлами»

Каждый узел, кроме последнего, связывается со следующим (посредством указателей)

Последний узел указывает на nullptr (0)

Самый первый узел называется «**головой»**.



Каждый элемент представляет собой структуру,

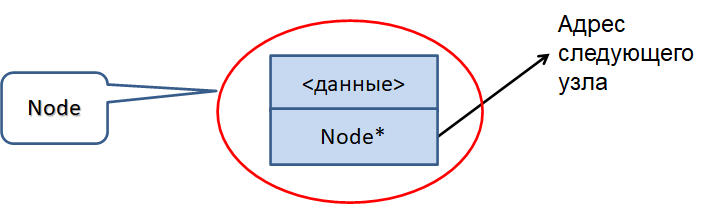
struct **Node**

{

< данные>

**Node** \* pNext; // **указатель** на следующий элемент

};



# Алгоритмы для создание и работы с односвязным списком

Рассмотрим некоторые алгоритмы, которые можно применить при работе с односвязным списком. Список этих алгоритмов будет далеко не полным.

## Создание узла односвязного списка

Node\* pCur = new Node;

pCur->data = value;

pCur->pNext = nullptr;

## Создание односвязного списка

Для того, чтобы можно было работать со списком, необходимо иметь указатель на начало списка, т.е. указатель на «голову».

Кроме указателя на начало списка полезно сохранять текущее число элементов

Первоначально в списке нет узлов, следовательно,

Node \* pHead =nullptr; //указатель на голову списка

size\_t size=0; // число элементов

## Добавление элементов в односвязный список

Новые элементы можно добавлять в любое место списка (в начало, в конец, в «середину»). Наиболее просто добавлять элементы в начало списка (в «голову»).

Для добавления элемента в конец списка требуется последовательно пройти по всему списку.

Для добавления в «середину» требуется иметь информацию о предыдущем узле.

## Добавление элементов в начало списка

1. Динамически cоздается «новый» элемент
2. Для «нового» элемента следующим становится тот узел, который прежде был «головой»
3. После этого «новый» элемент сам становится «головой»

В результате элементы списка располагаются в порядке, обратном порядку заполнения

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram, schematic

Description automatically generated

## Исключение из списка одного узла c заданным значением

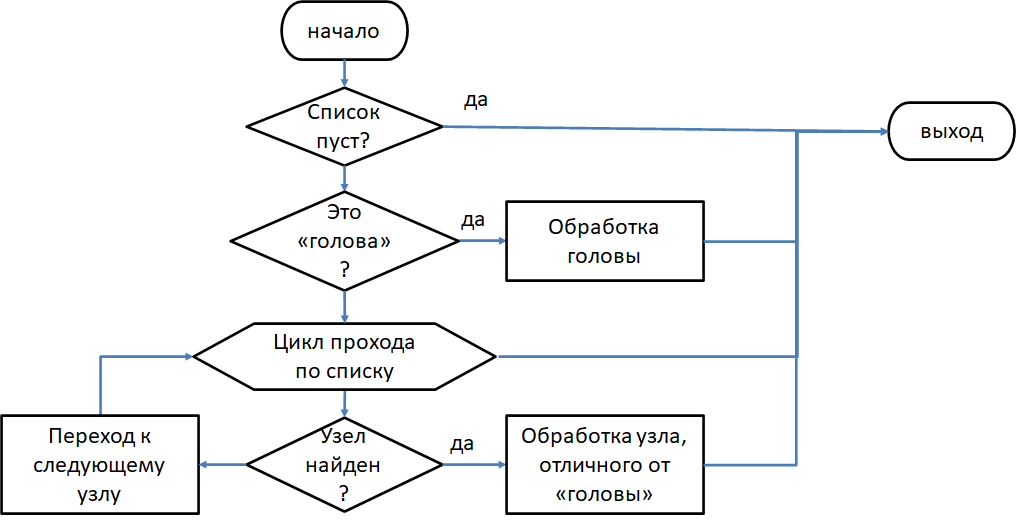
Рассмотрим вариант, когда удалить требуется первый узел, у которого значение данных совпадает с заданным.

При исключении узла из односвязного списка следует рассмотреть два случая.

Исключаемый узел:

* является «головой» списка
* не является «головой» списка

Один из алгоритмов может выглядеть так:



## Исключение узла из списка

1. Прежде всего необходимо проверить, что в списке имеются элементы
2. Если список не пуст, то проверим, является ли «голова» искомым узлом. Если да, то требуется удалить «голову».

Последовательность действий:

* Во временной переменной запомнить указатель на узел, следующий за «головой»

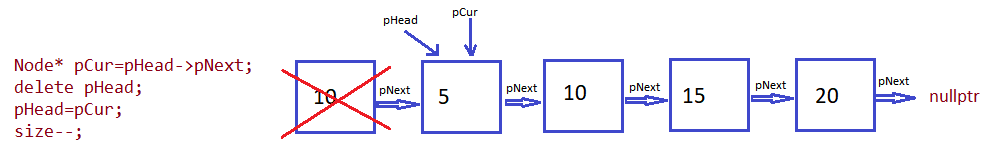
Node\* pCur= pHead->pNext;

* Удалить «голову»

delete pHead;

* установить новое значение для указатела на «голову» (pHead), чтобы можно было продолжать работать со списком.

pHead=pCur;



1. Если искомый узел не является «головой», то надо использовать цикл для прохода по списку и поиска нужного узла.

Для того, чтобы после удаления найденного узла в списке не образовались «дыры», можно запоминать указатель на ПРЕДЫДУЩИЙ (для удаляемого) элемент;

* Так как искомый узел не является «головой», то можно установить в качестве указателя на предыдущий элемент значение pHead.

Node\* pPrev=pHead;

* И начать циклический просмотр списка со следующего элемента.

Node\* pCur=pHead->pNext;

* В цикле, пока не дошли до конца списка поверяем не совпадает ли значение данных pCur->data с заданным значением.
* Если текущий узел является искомым, то требуется удалить текущий узел (на который указывает pCur).

Последовательность действий:

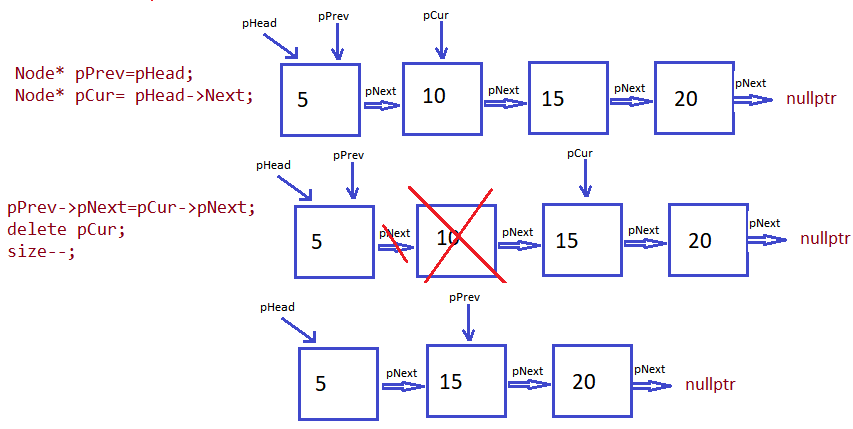
* Для предыдущего элемента изменить указатель на следующий элемент pPrev->pNext= pCur->pNext;
* Удалить текущий элемент

delete pCur;

* Если текущий узел не является искомым, то надо перейти к рассмотрению следующего элемента списка, а для этого изменить указатели pPrev и pCur.

pPrev=pCur;

pCur=pCur->pNext;



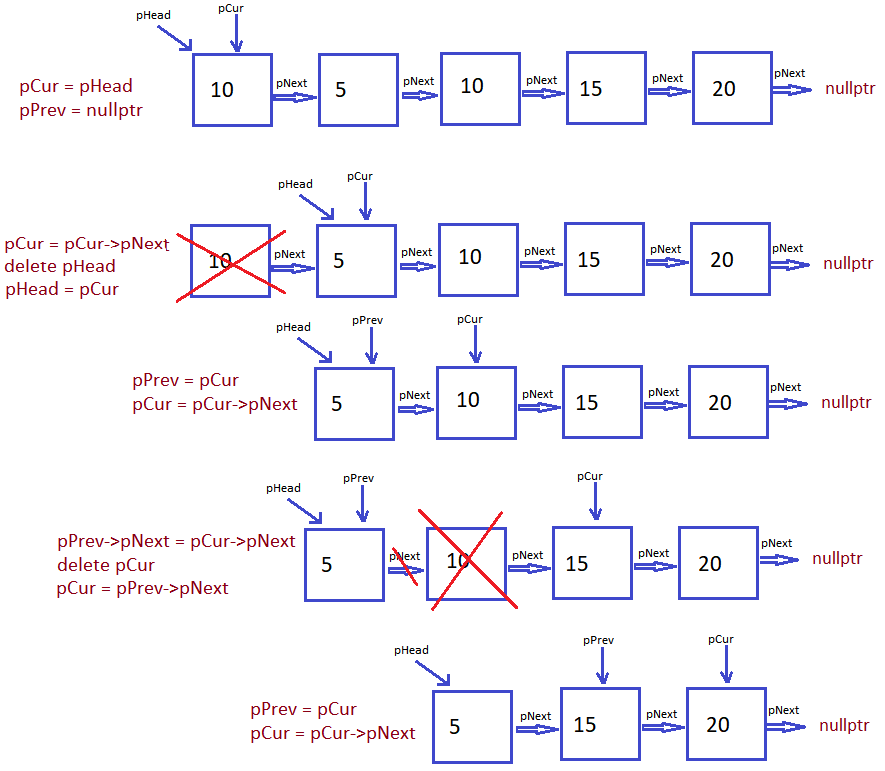
## Исключение из списка всех узлов c заданным значением

Над этим алгоритмом Вам предлагается подумать самим. При отладке этого алгоритма имеет смысл сформировать так список, чтобы:

* Несколько одинаковых элементов находились в начале списка;
* Несколько одинаковых элементов находились в конце списка;
* Несколько одинаковых элементов были разбросаны в середине списка.

Замечание: Не следует использовать для реализации этой функции вызов функции RemoveOne().

В качестве подсказки может служить следующая картинка:



# Реализация односвязного списка

Самым простым, но далеко не самым лучшим вариантом реализации односвязного списка является использование глобальных переменных, которые сохраняют значения «головы» списка и размер списка.

Можно создать файл Node.cpp, в который поместить определения глобальных данных и необходимых функций, а также файл Node.h, в котором будут находиться соответствующие объявления.

Node.h

Node.cpp

//ГЛОБАЛЬНЫЕ данные

Node \* pHead= nullptr;

size\_t size=0;

void AddToHead(int value) { . . . }

void PrintList() { . . . }

bool RemoveOne(int value) {. . .}

//ГЛОБАЛЬНЫЕ данные

extern Node \* pHead;

extern size\_t size;

void AddToHead(int value);

void PrintList();

bool RemoveOne(int value);

//------------------------------------------------------------------

// Добавление узла в голову списка

//------------------------------------------------------------------

void AddToHead(int value)

{

Node \* pCur = new Node ;

pCur->data=value;

pCur >pNext = pHead;

pHead = pCur;

size++;

}

//------------------------------------------------------------------

//Проход по списку - Печать списка в порядке, обратном заполнению

//------------------------------------------------------------------

void PrintList()

{

const Node\* ptmp = pHead;

while (ptmp != nullptr)

{

std::cout << ptmp->data << " ";

ptmp = ptmp->pNext;

}

std::cout << std::endl;

}

# Предпочтительная реализация односвязного списка

Удобно список представлять как структуру, в которую заключен указатель на начало списка и число элементов списка.

struct List

Diagram

Description automatically generated{

Node \* pHead; //указатель на голову списка

size\_t size; //число элементов

};

## Создание односвязного списка (предпочтительный вариант)

1. Объявить структуры Node и List;
2. Создать структурную переменную типа List;

List list1;

1. задать начальные значения структурной переменной (первоначально в списке нет узлов)

list1.pHead=nullptr;

list1.size=0;

1. Cоздать файл List.cpp, в который поместить определения необходимых функций, а также файл List.h, в котором будут находиться соответствующие объявления.

List.h

List.cpp

//реализации функций

void AddToHead(???) { . . . }

void PrintList(???) { . . . }

bool RemoveOne(???) {. . .}

. . .

//необходимые объявления

void AddToHead(???);

void PrintList(???);

bool RemoveOne(???);

. . .