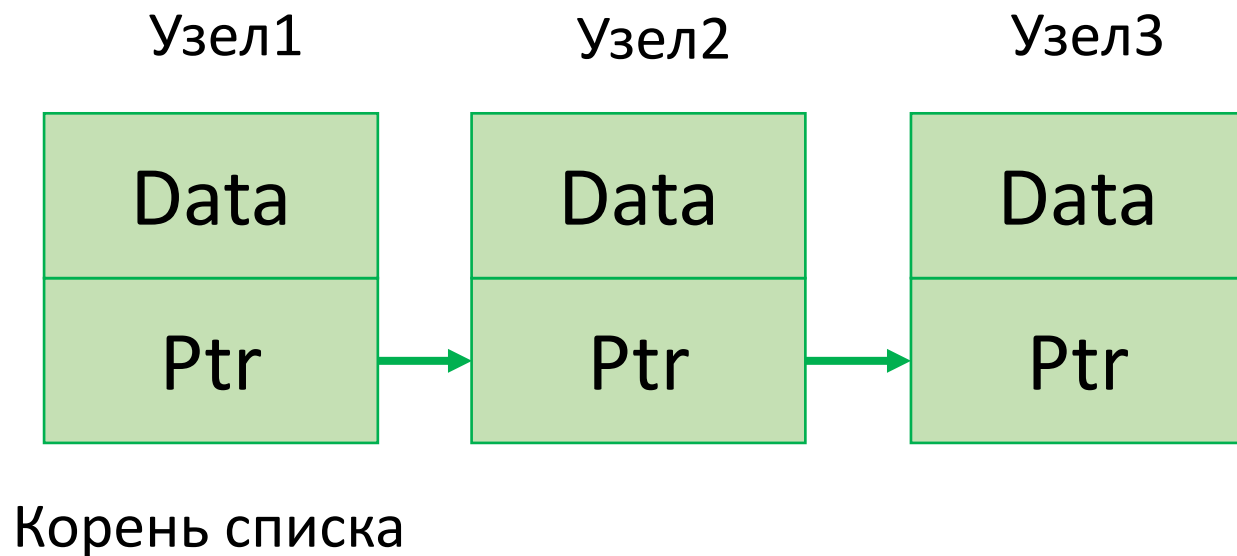




# Последовательные контейнеры STL



# Связные списки

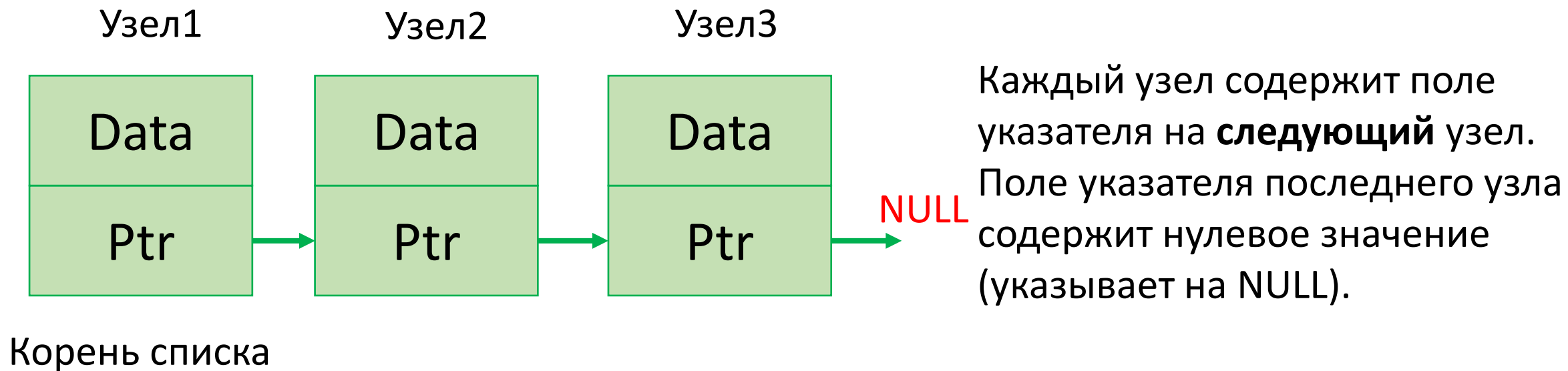


**Связный список** является простейшим типом данных динамической структуры, состоящей из элементов (узлов). Каждый узел включает в себя два поля:

- данные
- указатель на следующий узел в списке.



# Односвязные списки





# Односвязные списки: реализация

```
struct list
{
    int data;
    struct list* ptr;
};
```

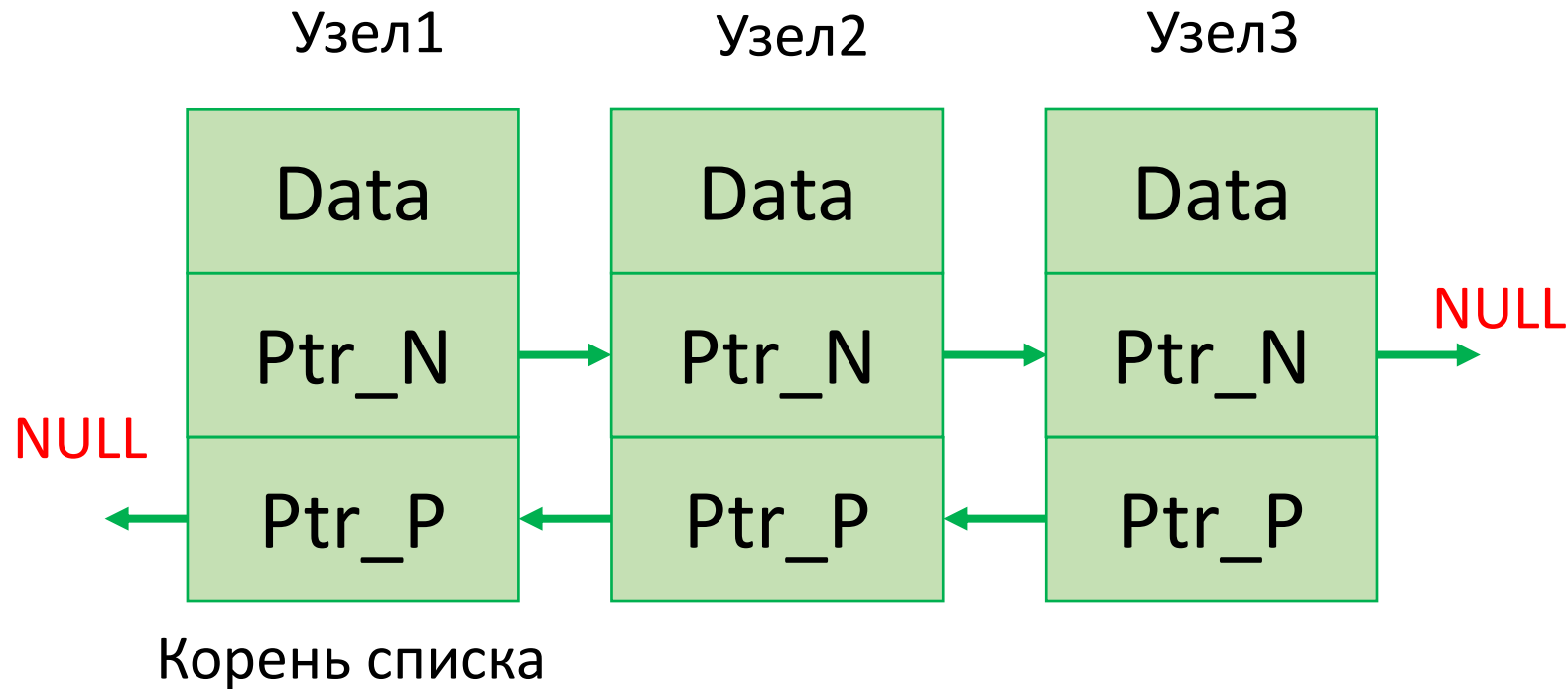
```
struct list* init_list(int a)
{
    struct list* lst;
    lst = (struct list*)malloc(sizeof(struct list));
    lst->data = a;
    lst->ptr = NULL;
    return lst;
}
```

Инициализация списка

Полный код см. в репозитории



# Двусвязные списки



Каждый узел содержит два поля указателей: на следующий и на предыдущий узел.

Поле указателя последнего и первого узла содержит нулевое значение (указывает на NULL).



# Двусвязные списки: реализация

```
struct list
{
    int data;
    struct list* next;
    struct list* prev;
};
```

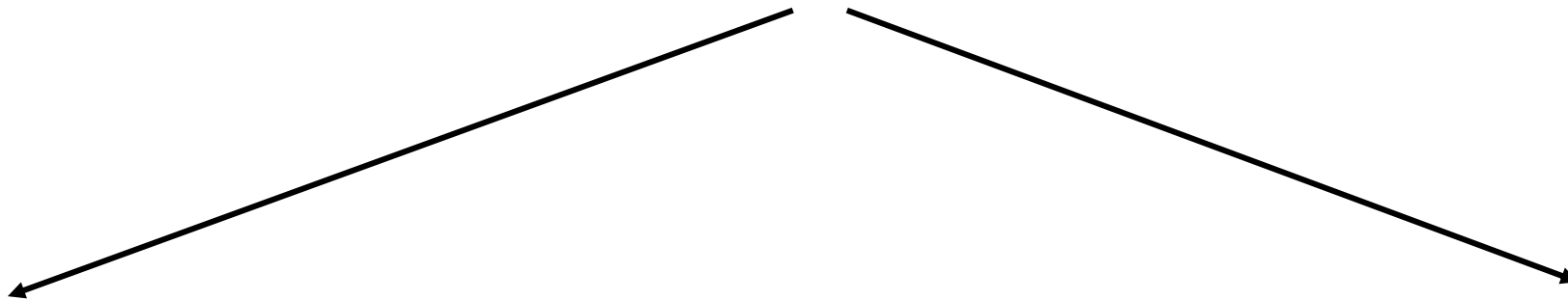
Инициализация списка

```
struct list* init_list(int a)
{
    struct list* lst = (struct list*)malloc(sizeof(struct list));
    lst->data = a;
    lst->next = NULL;
    lst->prev = NULL;
    return lst;
}
```

Полный код см. в репозитории



# Контейнеры STL



## Последовательные

array  
vector  
deque  
forward\_list  
list

## Ассоциативные

set  
map  
multiset  
multimap



# Std::array

```
#include <iostream>
#include <algorithm>
#include <array>

int main()
{
    srand(time(NULL));
    std::array<int, 10> arr;
    for (size_t i = 0; i < 10; ++i)
        arr[i] = rand() % 100;
    std::sort(arr.begin(), arr.end());
    for (auto it = arr.begin(); it != arr.end(); it++)
        std::cout << *it << std::endl;
    return 0;
}
```

Обертка над  
классическим  
**статическим C-массивом**





# Std::vector

```
#include <iostream>
#include <algorithm>
#include <vector>

int main()
{
    srand(time(NULL));
    std::vector<int> arr;
    for (size_t i = 0; i < 10; ++i)
        arr.push_back(rand() % 100);
    std::sort(arr.begin(), arr.end());
    for (auto it = arr.begin(); it != arr.end(); it++)
        std::cout << *it << std::endl;
    return 0;
}
```

Обертка над  
классическим  
**динамическим C-**  
массивом



# Std::deque

```
#include <iostream>
#include <algorithm>
#include <deque>
int N = 10;
int main()
{
    srand(time(NULL));
    std::deque<int> arr;
    for (size_t i = 0; i < N; ++i)
        arr.push_back(rand() % 100);
    std::sort(arr.begin(), arr.end());
    for (auto it = arr.begin(); it != arr.end(); it++)
        std::cout << *it << std::endl;
    return 0;
}
```

Возможность добавлять  
элементы в начало  
массива



# Std::forward\_list

```
#include <iostream>
#include <algorithm>
#include <forward_list>
int N = 10;
int main()
{
    srand(time(NULL));
    std::forward_list<int> arr;
    for (size_t i = 0; i < N; ++i)
        arr.push_front(rand() % 100);
    arr.sort();
    for (auto it = arr.begin(); it != arr.end(); it++)
        std::cout << *it << std::endl;
    return 0;
}
```

Односвязный список



# Std::list

```
#include <iostream>
#include <algorithm>
#include <list>
int N = 10;
int main()
{
    srand(time(NULL));
    std::list<int> arr;
    for (size_t i = 0; i < N; ++i)
        arr.push_front(rand() % 100);
    arr.sort();
    for (auto it = arr.begin(); it != arr.end(); it++)
        std::cout << *it << std::endl;
    return 0;
}
```

Двусвязный список

Полный код см. в репозитории