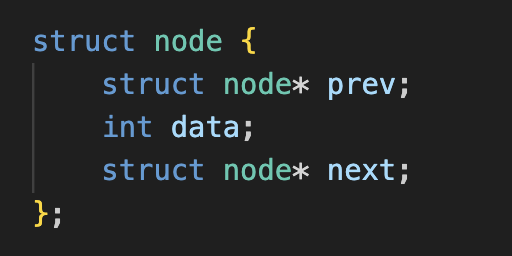
**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на тему «Двусвязный список» (шестое задание во втором семестре)**

Односвязный список отличается от двусвязного наличием у последнего дополнительного указателя, который «указывает» на предыдущий элемент списка.

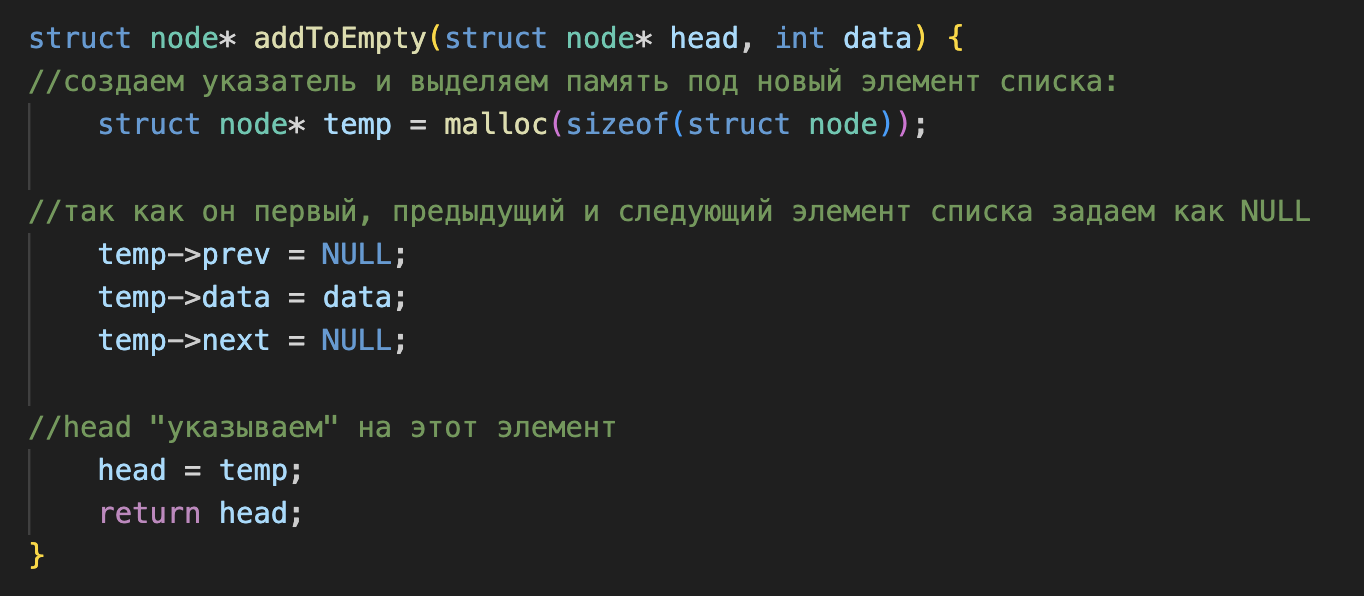
Соответственно структура, отвечающая за хранение элемента списка, теперь будет выглядеть так:



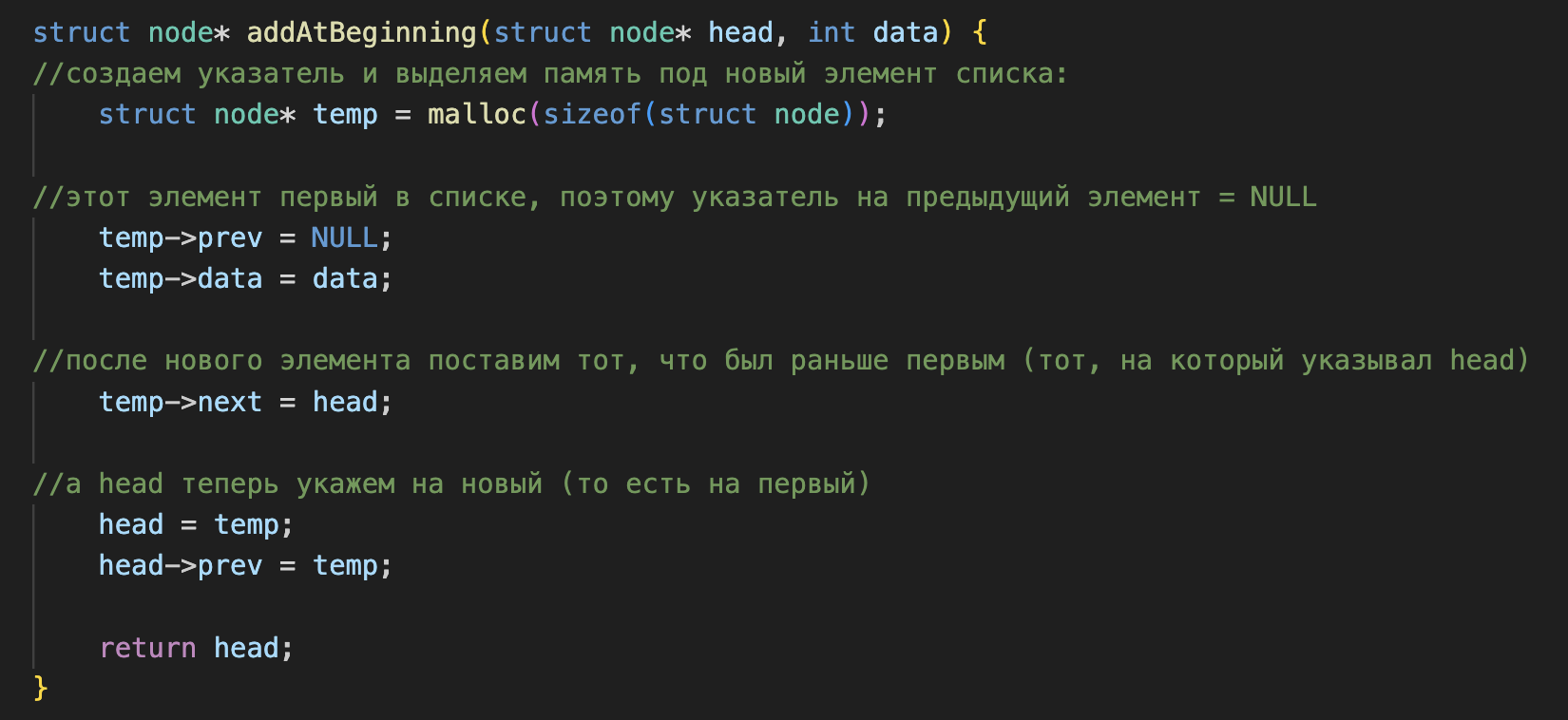
Для освоения темы я решил написать все основные функции, необходимые для работы с двусвязными списками.

Важное замечание: в то время как при работе с односвязными списками я использовал *указатели на указатели* при передаче данных в функции (это позволяло создавать функции, которым ничего не нужно возвращать), в этой работе я решил опробовать другой метод. Теперь я буду передавать в функцию *указатель на начало*, производить в ней с его помощью изменения в списке (тем самым зачастую меняя *указатель на начало*) и возвращать новый *указатель на начало*.

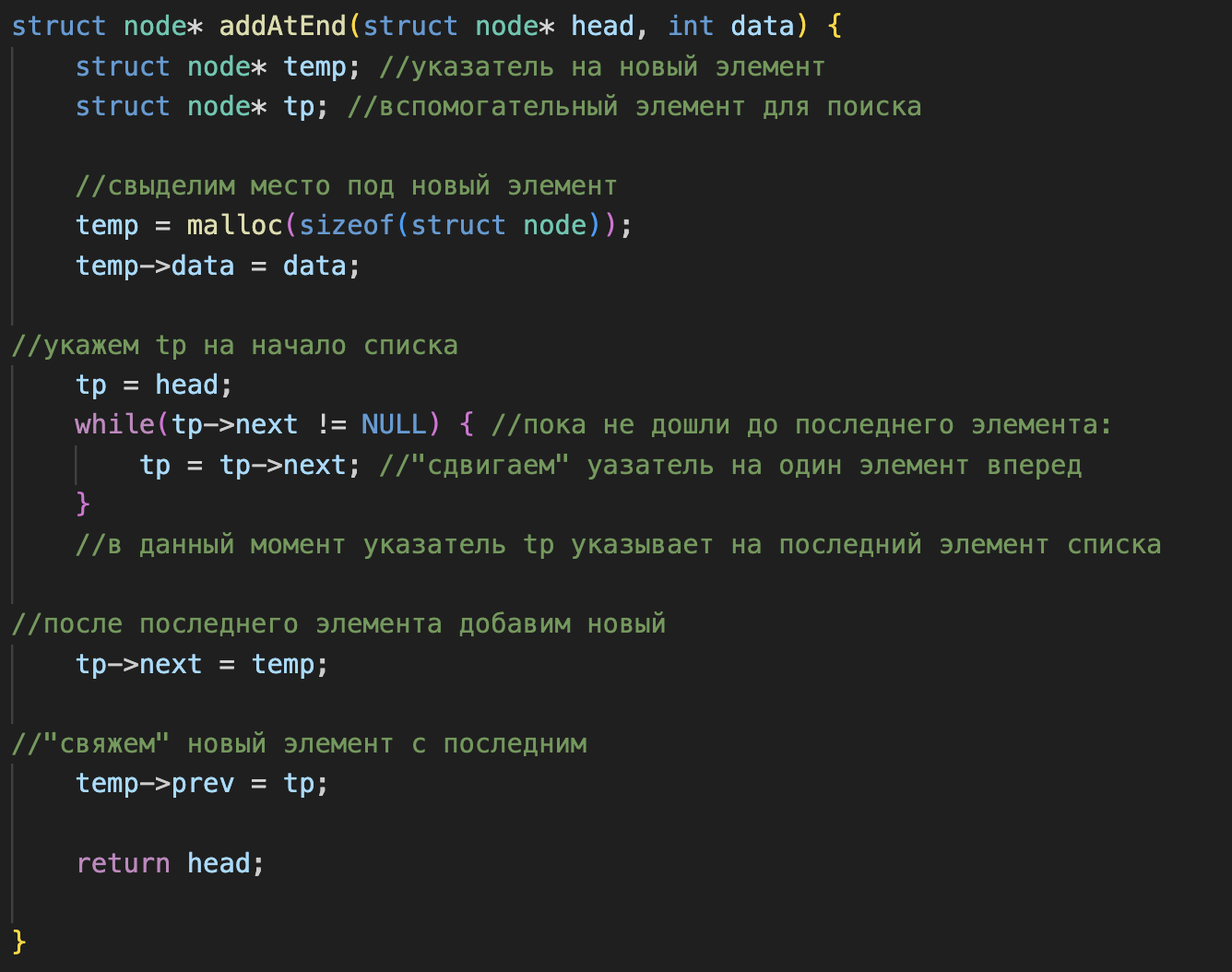
1) **Добавление элемента в пустой список**



2**) Добавление элемента в начало**

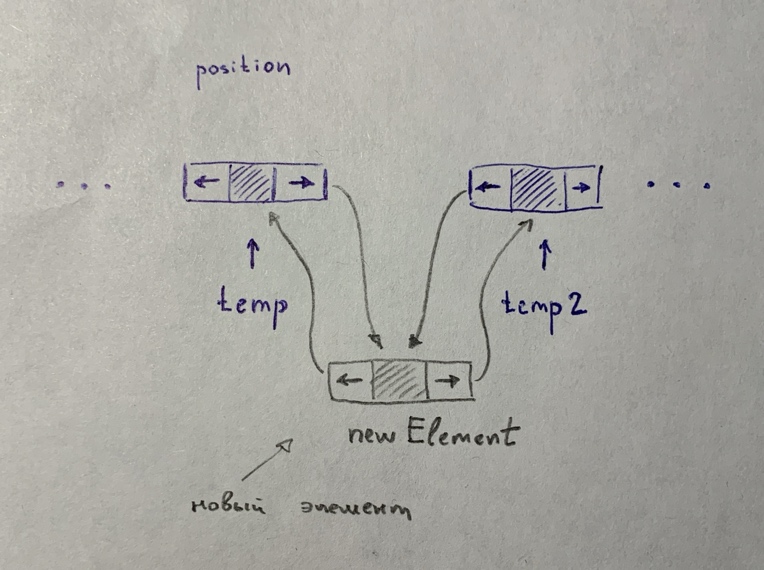


3) **Добавление элемента в конец**

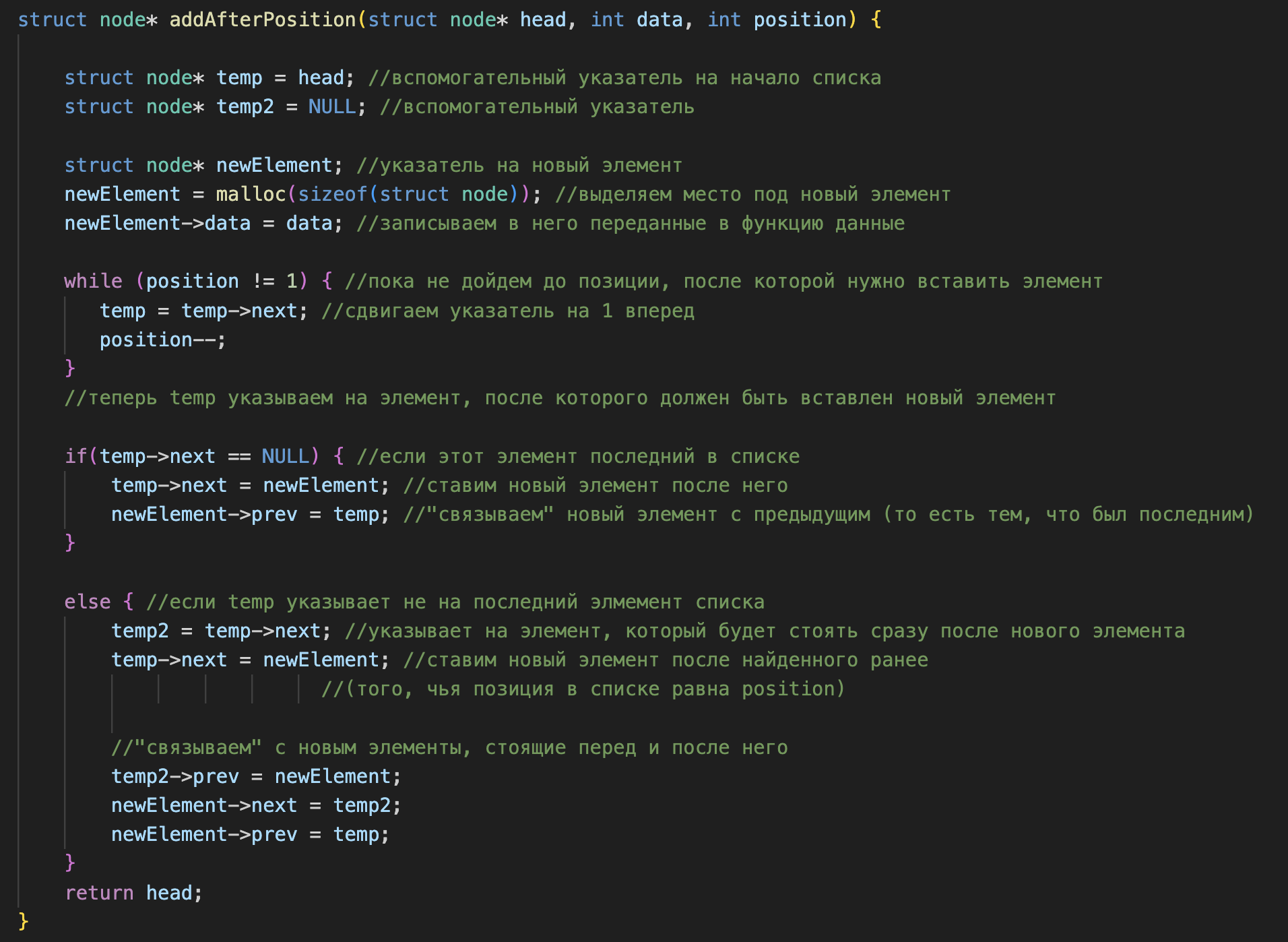


4) **Добавление элемента после определенной позиции**

Здесь я схематично изобразил механизм вставки:

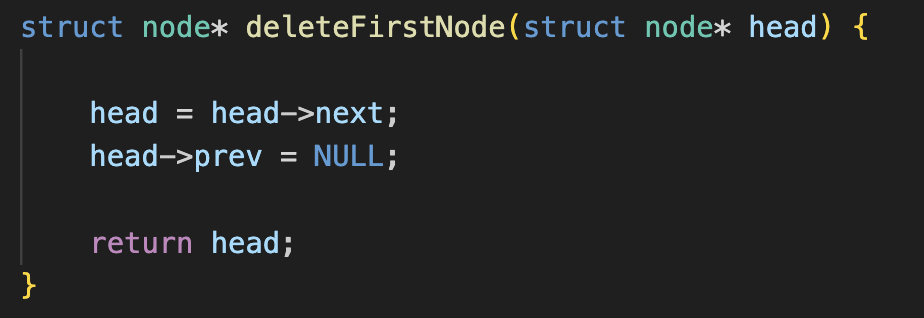


Код:

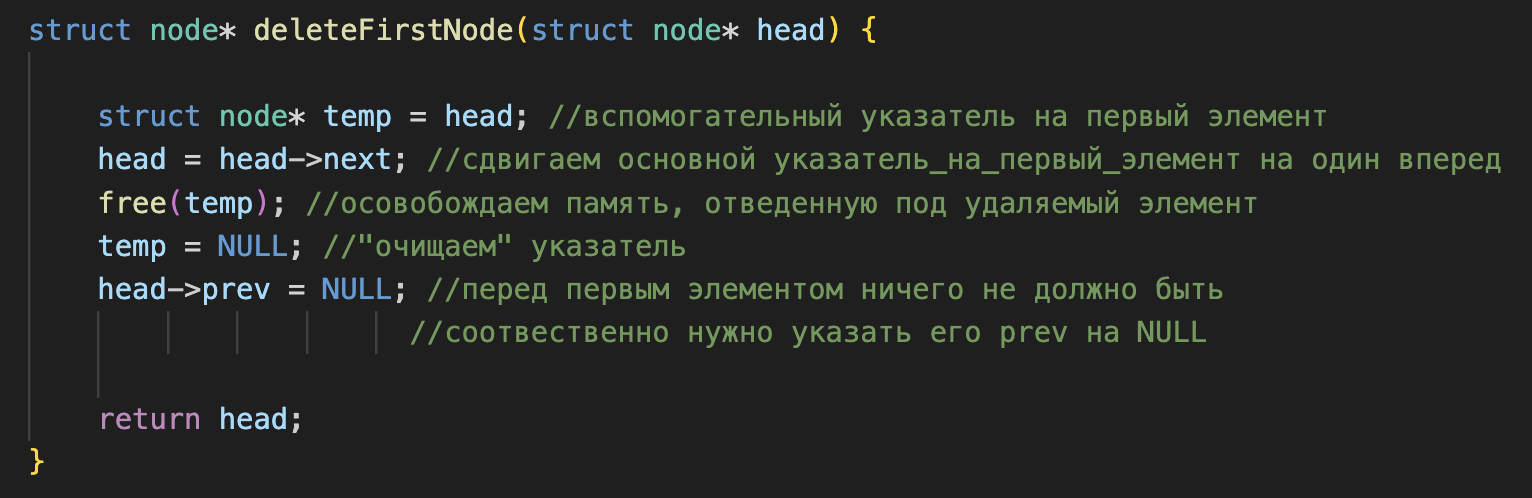


5) **Удаление первого элемента**

Можно было бы сделать вообще элементарно:

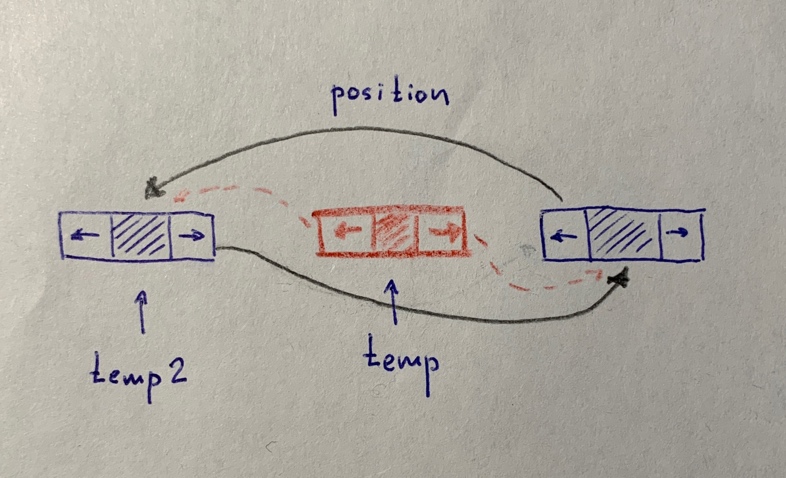


Но при таком подходе мы игнорируем тот факт, что «удаленный» элемент все еще занимает место. Чтобы решить эту проблему придется немного усложнить код:

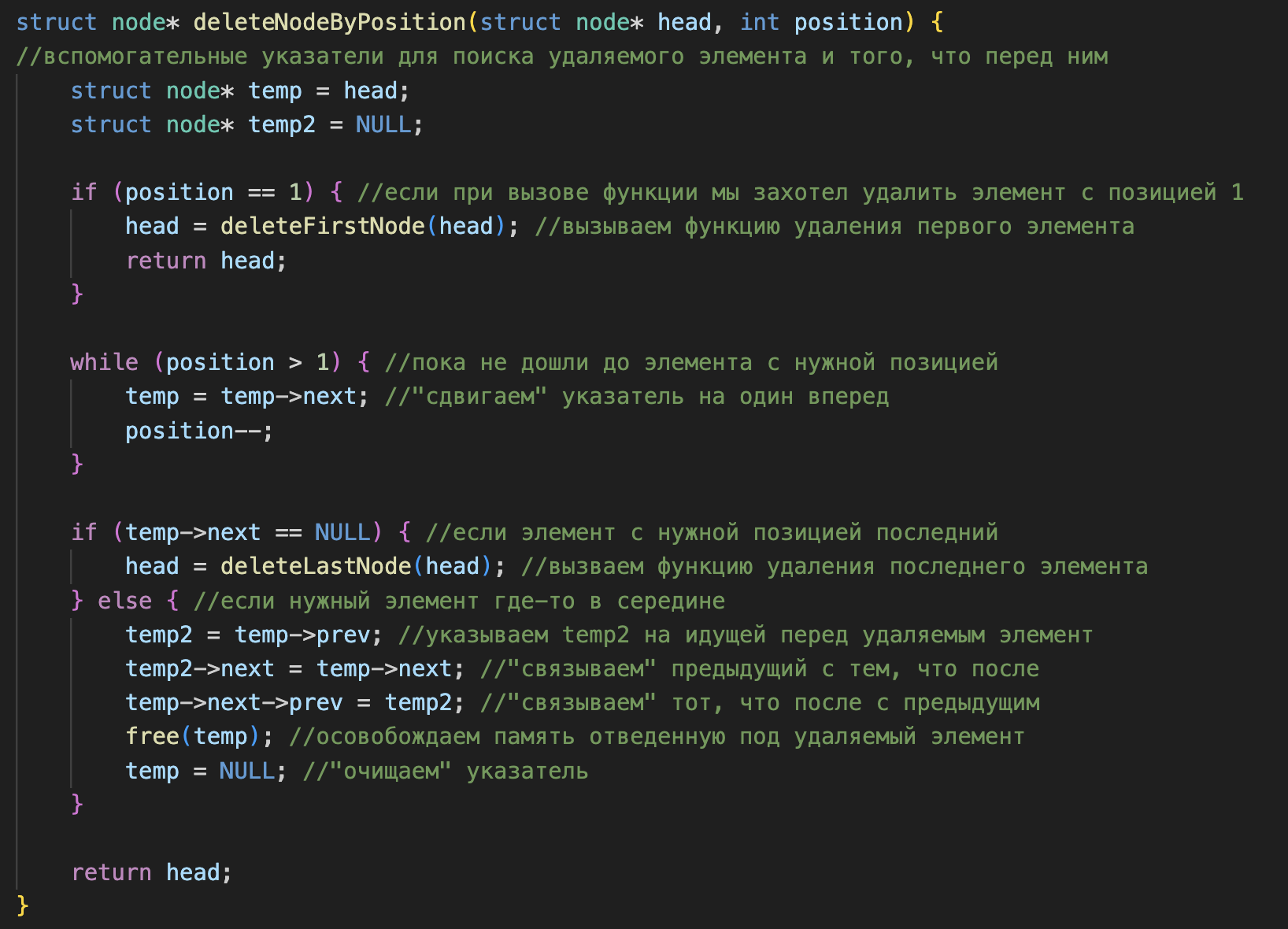


6) **Удаление элемента по позиции**

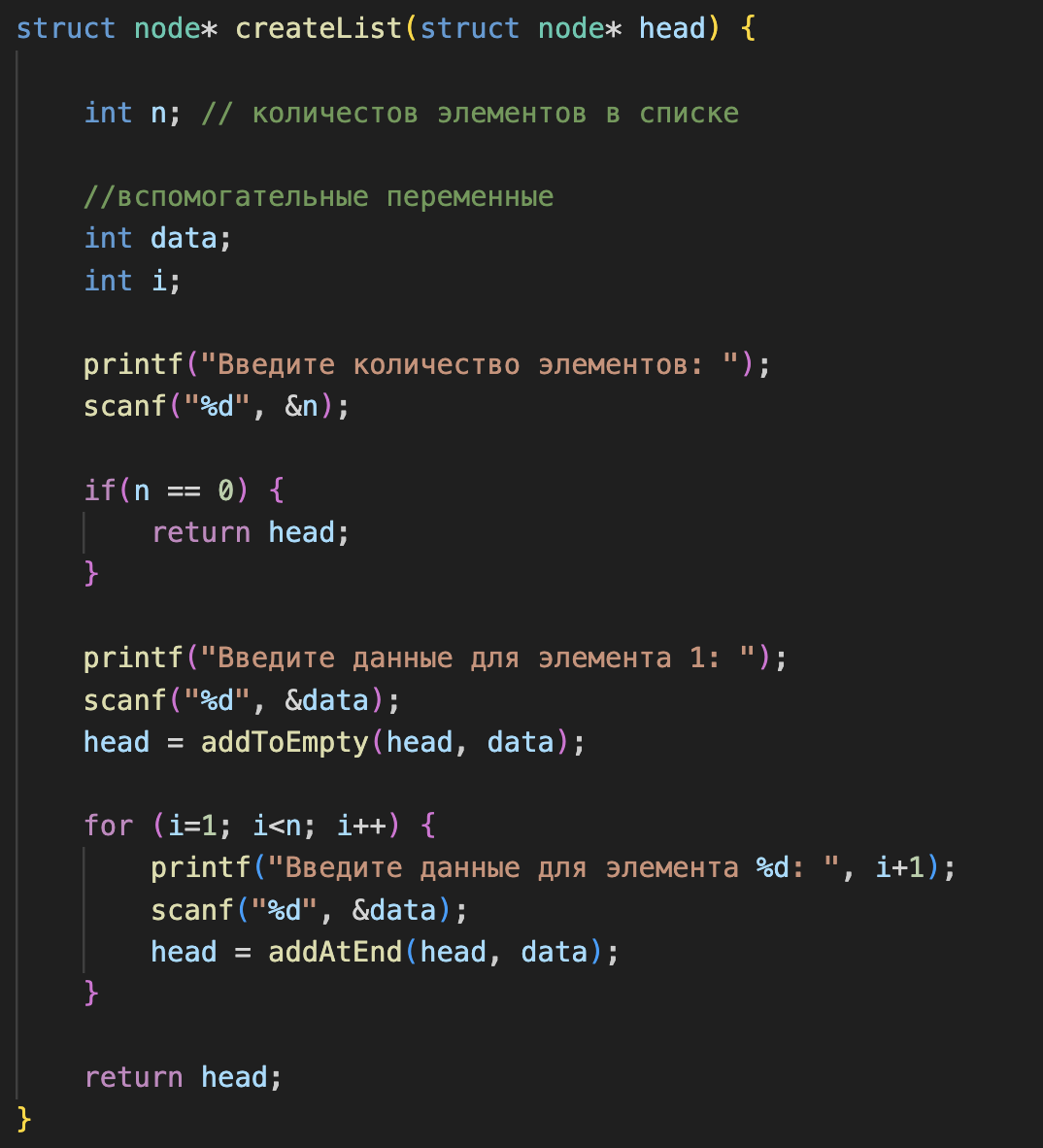
Схема:



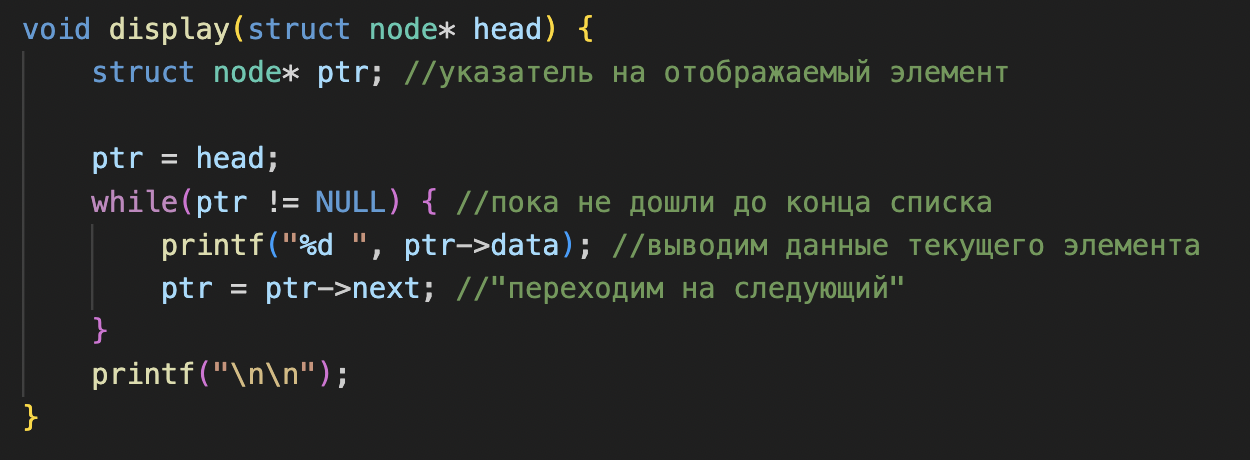
Код:



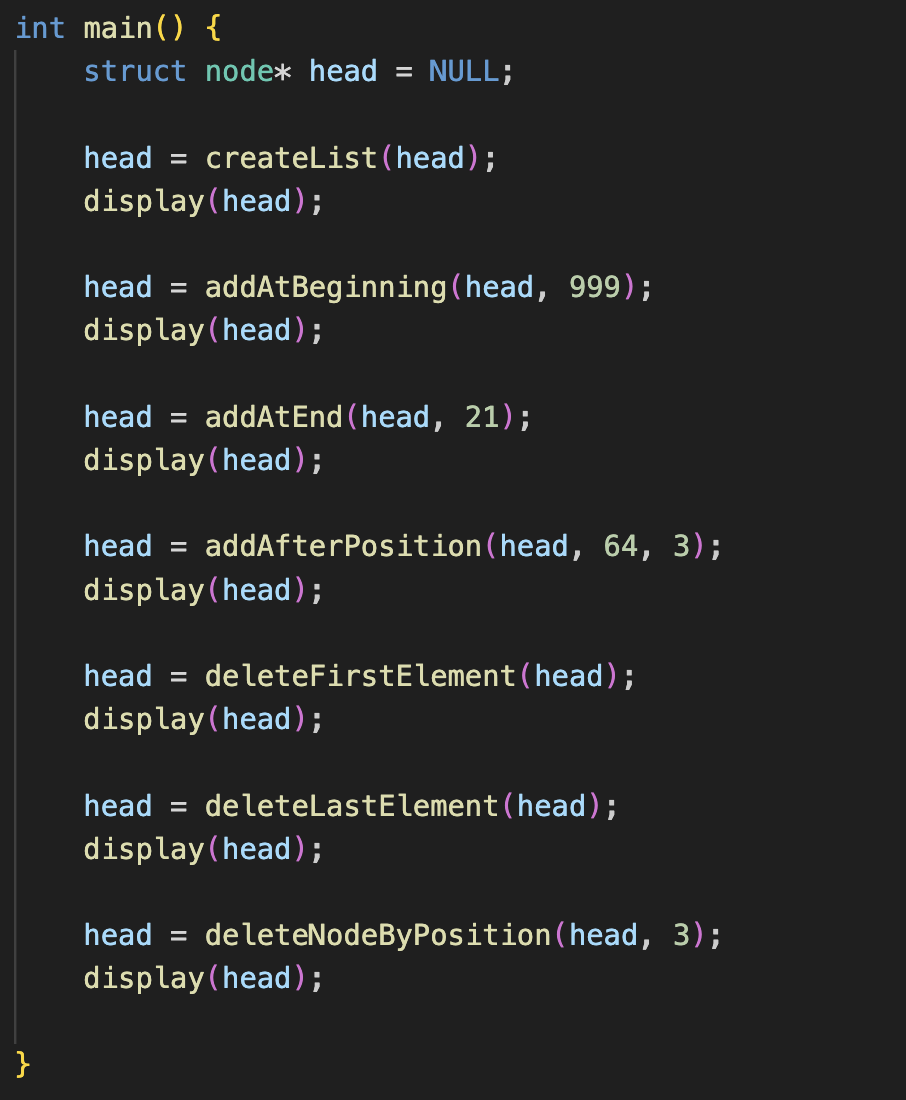
7) **Формирование начального списка**



8) **Отображение всего списка**

****

**Проведем тест программы:**

****

В терминале:

