Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №10**

Дисциплина: Информатика

Тема: Динамические массивы

Выполнил работу

студент группы ИВТ-20-1б

Михайлов М. А.

Проверила

Доцент кафедры ИТАС

к.т.н Полякова О. А.

Пермь, 2021

**Цель работы**

Получить практические навыки работы с двунаправленными списками.

**Постановка задачи**

(Вариант 19)

Тип информационного поля char\*.

Добавить в список элементы с номерами 1, 3, 5 и т. д.

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать структуру node, поля которой будут использоваться для ввода данных типа char и для хранения адресов предыдущего и следующего элемента.

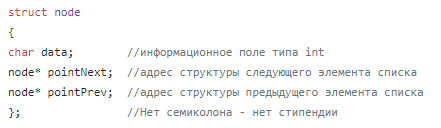
**1.2.** Разработать функцию make для формирования списка.

**1.**3. Разработать функцию print для печати списка.

**1.**4. Разработать функцию add\_1\_3\_5 для добавления элементов с номерами 1, 3, 5 и т.д.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Структура node для организации двунаправленного списка.



**2.2.** Для функции make используются следующие аргументы:

1. Тип int – размер списка.

**2.3.** Для функции print используются следующие аргументы:

1. Указатель на элемент массива first – в функции происходит проверка на указание на null.

Сама функция имеет тип void, так как ничего не надо возвращать – функция только выводит список в консоль.

**2.4.** Для функции add\_1\_3\_5 используются следующие аргументы:

1. Указатель на ссылку на first – указывается первый элемент в списке в качестве параметра.

Сама функция имеет тип void, возвращать ничего не надо.

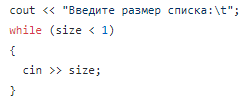
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через консоль в виде двунаправленного списка через структуру (тип node и полями типа char).

**4.** Для функции ввода и вывода использовались следующие операторы и функции:

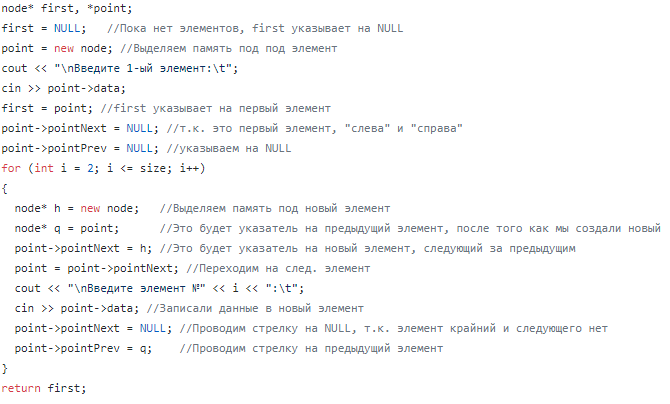
**4.1.** Вывод данных на консоль реализован с помощью cout.

**4.2.** Ввод данных с консоли реализован через cin, который стоит внутри цикла while, который проверяет размер и повторяется пока размер списка не будет ниже 1.

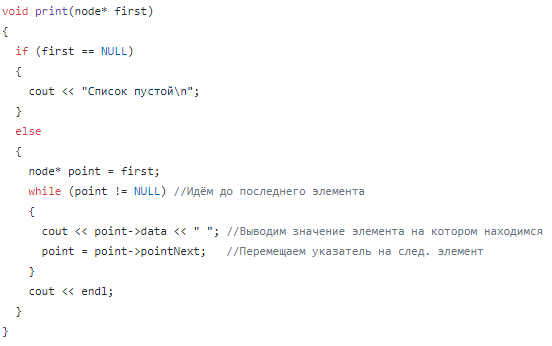


**5.** Поставленная задача будет решена следующими действиями:

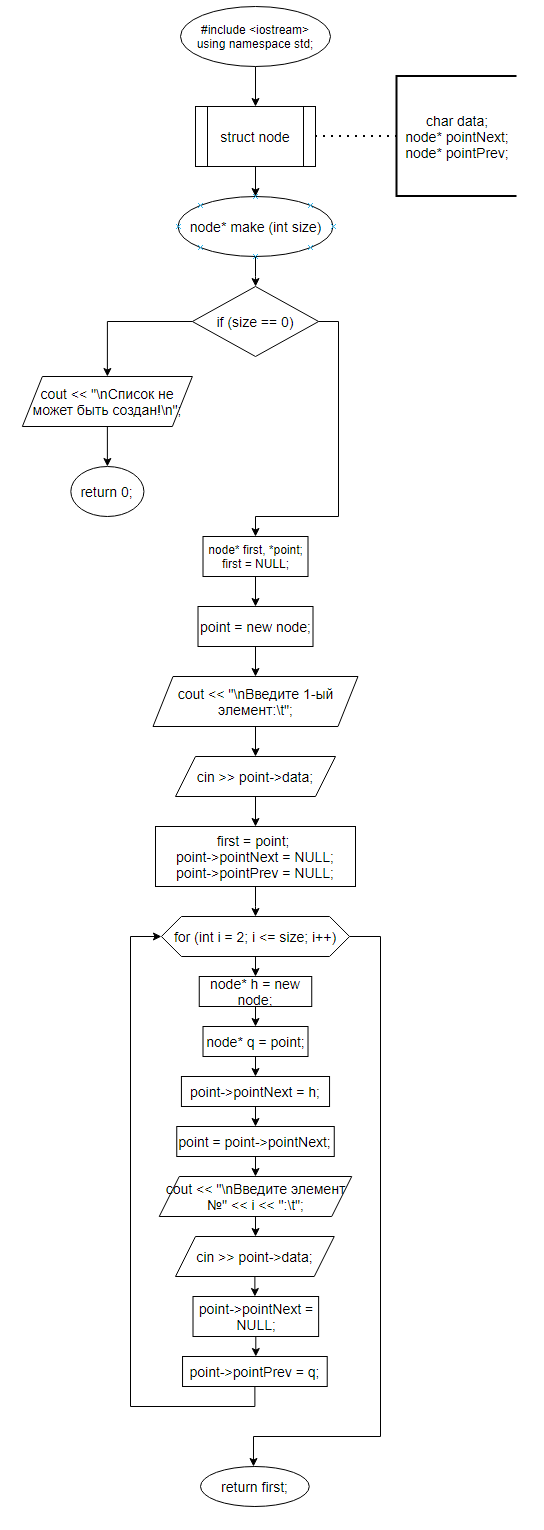
**5.1.**  Ввод символов в список происходит внутри функции make. Первый вводится вне цикла for и используется в качестве указателя на список. В цикле for заполняются элементы от 2 до size включительно. Выделяется память переменной h под новый элемент, следующий за предыдущим. Предыдущий элемент будет сохранён в pointPrev, а следующий будет сохранён в pointNext.

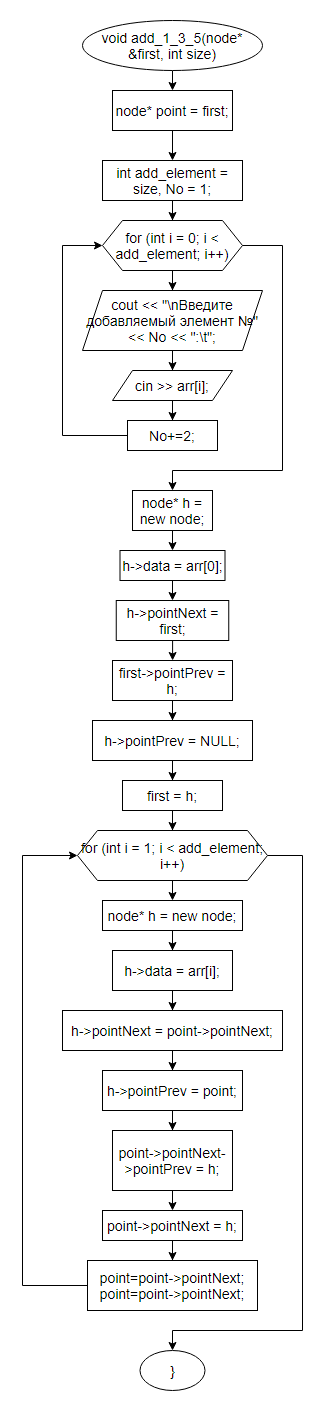
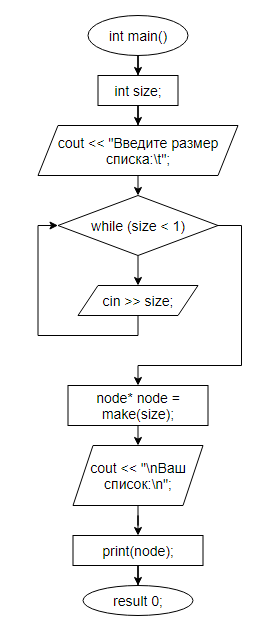


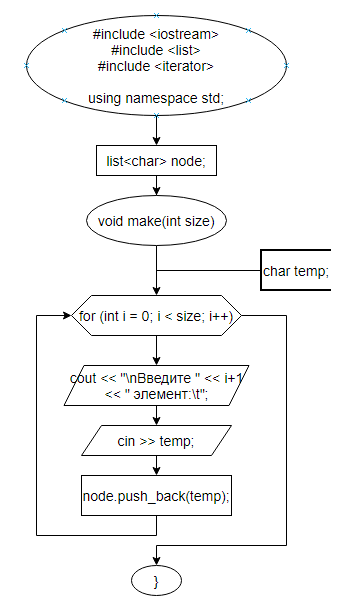
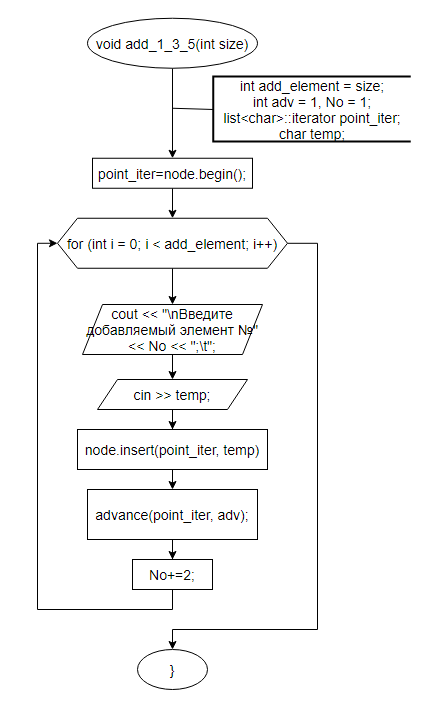
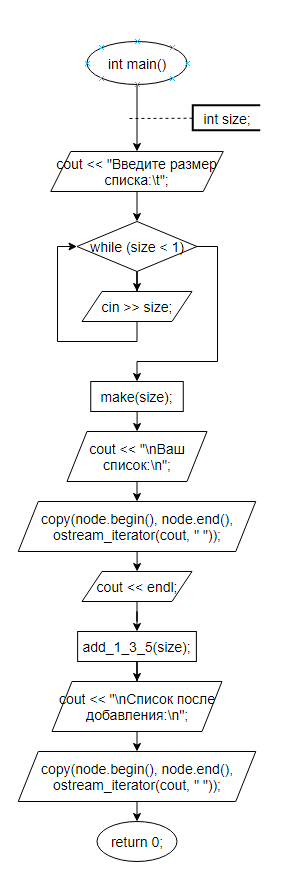
**5.2.** Вывод списка на консоль реализован с помощью функции print.



**Блок-схема**

****

**** ****

**** **** 

**Код программы**

**Структуры**

|  |
| --- |
| #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | struct node |
|  | { |
|  | char data; //информационное поле типа int |
|  | node\* pointNext; //адрес структуры следующего элемента списка |
|  | node\* pointPrev; //адрес структуры предыдущего элемента списка |
|  | }; //Нет семиколона - нет стипендии |
|  |  |
|  | node\* make (int size) |
|  | { |
|  | if (size == 0) |
|  | { |
|  | cout << "\nСписок не может быть создан!\n"; |
|  | return 0; |
|  | } |
|  | node\* first, \*point; |
|  | first = NULL; //Пока нет элементов, first указывает на NULL |
|  | point = new node; //Выделяем память под под элемент |
|  | cout << "\nВведите 1-ый элемент:\t"; |
|  | cin >> point->data; |
|  | first = point; //first указывает на первый элемент |
|  | point->pointNext = NULL; //т.к. это первый элемент, "слева" и "справа" |
|  | point->pointPrev = NULL; //указываем на NULL |
|  | for (int i = 2; i <= size; i++) |
|  | { |
|  | node\* h = new node; //Выделяем память под новый элемент |
|  | node\* q = point; //Это будет указатель на предыдущий элемент, после того как мы создали новый |
|  | point->pointNext = h; //Это будет указатель на новый элемент, следующий за предыдущим |
|  | point = point->pointNext; //Переходим на след. элемент |
|  | cout << "\nВведите элемент №" << i << ":\t"; |
|  | cin >> point->data; //Записали данные в новый элемент |
|  | point->pointNext = NULL; //Проводим стрелку на NULL, т.к. элемент крайний и следующего нет |
|  | point->pointPrev = q; //Проводим стрелку на предыдущий элемент |
|  | } |
|  | return first; |
|  | } |
|  |  |
|  | void print(node\* first) |
|  | { |
|  | if (first == NULL) |
|  | { |
|  | cout << "Список пустой\n"; |
|  | } |
|  | else |
|  | { |
|  | node\* point = first; |
|  | while (point != NULL) //Идём до последнего элемента |
|  | { |
|  | cout << point->data << " "; //Выводим значение элемента на котором находимся |
|  | point = point->pointNext; //Перемещаем указатель на след. элемент |
|  | } |
|  | cout << endl; |
|  | } |
|  | } |
|  |  |
|  | void add\_1\_3\_5(node\* &first, int size) |
|  | { |
|  | node\* point = first; |
|  | int add\_element = size, No = 1; |
|  | char arr[add\_element]; |
|  | for (int i = 0; i < add\_element; i++) |
|  | { |
|  | cout << "\nВведите добавляемый элемент №" << No << ":\t"; |
|  | cin >> arr[i]; |
|  | No+=2; |
|  | } |
|  | node\* h = new node; |
|  | h->data = arr[0]; |
|  | h->pointNext = first; |
|  | first->pointPrev = h; |
|  | h->pointPrev = NULL; |
|  | first = h; |
|  |  |
|  | for (int i = 1; i < add\_element; i++) |
|  | { |
|  | node\* h = new node; |
|  | h->data = arr[i]; |
|  | h->pointNext = point->pointNext; |
|  | h->pointPrev = point; |
|  | point->pointNext->pointPrev = h; |
|  | point->pointNext = h; |
|  |  |
|  | point=point->pointNext; |
|  | point=point->pointNext; |
|  | } |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | setlocale (LC\_ALL, "ru"); |
|  | int size; |
|  | cout << "Введите размер списка:\t"; |
|  | while (size < 1) |
|  | { |
|  | cin >> size; |
|  | } |
|  |  |
|  | node\* node = make(size); |
|  | cout << "\nВаш список:\n"; |
|  | print(node); |
|  |  |
|  | add\_1\_3\_5(node, size); |
|  |  |
|  | cout << "\nСписок после добавления:\n"; |
|  | print(node); |
|  |  |
|  | return 0; |
|  | } |

**Библиотеки STL**

|  |
| --- |
| /\* |
|  | 19. Тип информационного поля char\*. |
|  | Добавить в список элементы с номерами 1, 3, 5 и т. д. |
|  | \*/ |
|  |  |
|  | #include <iostream> |
|  | #include <list> |
|  | #include <iterator> |
|  |  |
|  | using namespace std; |
|  |  |
|  | list<char> node; |
|  |  |
|  | void make(int size) |
|  | { |
|  | char temp; |
|  | for (int i = 0; i < size; i++) |
|  | { |
|  | cout << "\nВведите " << i+1 << " элемент:\t"; |
|  | cin >> temp; |
|  | node.push\_back(temp); //Вводим элемент в список с конца |
|  | } |
|  | } |
|  |  |
|  | void add\_1\_3\_5(int size) |
|  | { |
|  | int add\_element = size; |
|  | int adv = 1, No = 1; |
|  | list<char>::iterator point\_iter; |
|  | point\_iter=node.begin(); |
|  | char temp; |
|  | for (int i = 0; i < add\_element; i++) |
|  | { |
|  | cout << "\nВведите добавляемый элемент №" << No << ";\t"; |
|  | cin >> temp; |
|  | node.insert(point\_iter, temp); |
|  | advance(point\_iter, adv); |
|  | No+=2; |
|  | } |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | setlocale (LC\_ALL, "ru"); |
|  | int size; |
|  | cout << "Введите размер списка:\t"; |
|  | while (size < 1) |
|  | { |
|  | cin >> size; |
|  | } |
|  | make(size); |
|  | cout << "\nВаш список:\n"; |
|  | copy(node.begin(), node.end(), ostream\_iterator<char>(cout, " ")); |
|  | cout << endl; |
|  |  |
|  | add\_1\_3\_5(size); |
|  | cout << "\nСписок после добавления:\n"; |
|  | copy(node.begin(), node.end(), ostream\_iterator<char>(cout, " ")); |
|  |  |
|  | return 0; |
|  | } |