Лабораторная работа №3

std::list

Что изучим сегодня:

- std::list;
- итераторы;
- некоторые методы класса list.

std::list — это реализация двусвязного списка в C++. Список представляет собой контейнер, который поддерживает быструю вставку и удаление элементов из любой позиции в контейнере. Быстрый произвольный доступ не поддерживается. То есть, у list нет оператора [], что не позволит обратиться к любому произвольному элементу списка. Для прямого обращения к элементам списка предусмотрены два метода: `front()` и `back()`. Эти метода возвращают первый и последний элементы списка.

Для доступа к элементам списка используются итераторы. Итератор по своему синтаксису напоминает указатель на элемент списка. При инкременте итератора происходит переход этого итератора на следующий по порядку элемент списка.

Итераторы обеспечивают доступ к элементам контейнера. С помощью итераторов очень удобно перебирать элементы. Итератор описывается типом iterator. Но для каждого контейнера конкретный тип итератора будет отличаться. Так, итератор для контейнера `list<int>` представляет тип list<int>::iterator, а итератор контейнера `vector<int>` представляет тип vector<int>::iterator и так далее.

Для получения итераторов контейнеры в C++ обладают такими функциями, как begin() и end(). Функция begin() возвращает итератор, который указывает на первый элемент контейнера (при наличии в контейнере элементов). Функция end() возвращает итератор, который указывает на следующую позицию после последнего элемента, то есть по сути на конец

контейнера. Если контейнер пуст, то итераторы, возвращаемые обоими методами begin и end совпадают. Если итератор begin не равен итератору end, то между ними есть как минимум один элемент.

Итератор для конкретного типа контейнера:

```
std::list<int> v = { 1,2,3,4 };
std::list<int>::iterator iter = v.begin();  // получаем итератор
```

В данном случае создается вектор - контейнер типа list, который содержит значения типа int. Этот контейнер инициализируется набором {1, 2, 3, 4}. Через метод `begin()` можно получить итератор для этого контейнера, причем этот итератор будет указывать на первый элемент контейнера.

Вывод списка на экран через итераторы можно записать так:

```
void print(list<int> & a)
{
    //Обход массива с помощью итератора
    for (list<int>::iterator it = a.begin(); it!=a.end(); ++it)
    {
        cout<<(*it);
    }
}
```

Однако, в C++11 добавлена возможность упрощённого синтаксиса for, это позволяет сократить код:

```
//Печать массива с новым способом прохода массива
void print_cool(list<int> & a)
{
    for (int v: a)
    {
        cout<<v;
    }
}
```

Операции с итераторами

С итераторами можно проводить следующие операции:

- *iter: получение элемента, на который указывает итератор
- ++iter: перемещение итератора вперед для обращения к следующему элементу
- --iter: перемещение итератора назад для обращения к предыдущему элементу. Итераторы контейнера forward_list не поддерживают операцию декремента.
- iter1 == iter2: два итератора равны, если они указывают на один и тот же элемент
- iter1 != iter2: два итератора не равны, если они указывают на разные элементы

Задача 1. Ввести из консоли числа и записать их в список. Процесс ввода чисел должен закончиться при вводе 0. При этом сам 0 не добавляется в список. В процессе ввода числа должны добавляться в конец списка. Вывести числа в списке в порядке ввода, сделав функцию print()

Задача 2. Ввести из консоли числа, среди которых последнее равно 0. Сохранять 0 в список не надо.

а) Вывести только неповторяющиеся числа;

б) Отсортировать список;

в) Удалить из списка все двойки;

г) Удалить из списка все чётные элементы.

При выполнении этого задания используйте следующие функции класса std::list:

```
unique()- удаление повторяющихся элементов
sort()- сортировка элементов списка
remove(x) – удаление всех элементов, значение которых равно x.
erase(it)- удаление элемента по итератору it.
```

Важно!

При удалении элемента итератор на удалённый элемент становится некорретным, по этой причине функция erase() возвращает новый итератор. Это итератор на элемент, стоящий после удалённого элемента. Этот факт необходимо учитывать при выполнении задания 2 под буковой «в».