Семинар #5: Функциональные объекты.

Часть 1: Указатели на функции в алгоритмах STL

Функциональный объект – это объект, к которому можно применить оператор вызова функции (`operator()`). Простейшими функциональными объектами, введёными еще в языке `C`, являются указатели на функции. Многие алгоритмы STL могут принимать указатели на функции в качестве одного из параметров

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

int cmp(int a, int b) {
    return a > b;
}

int add_one(int a) {
    return a + 1;
}

int main() {
    std::vector v<int> {18, 51, 2, 25, 14, 97, 73};
    std::sort(v.begin(), v.end(), cmp);
    for (int num : v) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    std::transform(v.begin(), v.end(), v.begin(), add_one);
    for (int num : v) {
        std::cout << num << " ";
    }
}
```

В данном коде используются 2 алгоритма STL:

- `std::sort` третьим объектом принимает функциональный объект - компаратор.
- `std::transform` принимает на вход сначала 3 итератора. Первые два итератора задают последовательность объектов. Третий итератор указывает место куда нужно сохранить результат трансформации. В данном случае записывается тот же вектор, откуда берутся числа. Четвёртый аргумент – это функциональный объект, указывающий какую трансформацию нужно применить.

Задачи:

- Изменить код программы выше так, чтобы сортировка проходила по последней цифре.
- Изменить код программы выше так, функция `std::transform` увеличивала все числа в 2 раза.
- В файле `books.cpp` лежит заготовка, содержащая массив структур. Отсортируйте этот массив структур по возрастанию цены и напечатайте.
- Отсортируйте массив `books` по названию и напечатайте его.
- Увеличьте цену каждой книге на 10 процентов, используя функцию `std::transform`.

Часть 2: Функторы

Другим видом функциональных объектов в C++ является функтор. Функторы – это просто объекты класса, у которого есть перегруженный оператор `operator()`.

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

struct AddFunctor {
private:
    int x;
public:
    AddFunctor(int x) : x(x) {};

    operator()(int a) {
        return a + x;
    }
};


int main() {
    std::vector v<int> {18, 51, 2, 25, 14, 97, 73};

    std::transform(v.begin(), v.end(), v.begin(), AddFunctor(5));
    for (int num : v) {
        std::cout << num << " ";
    }
}
```

Задачи:

- Напишите `ModuloCmpFunctor`, который будет принимать в конструкторе некоторое число и в последствии этот функтор должен использоваться для передачи в функцию `std::sort`.
- Используйте этот функтор, чтобы отсортировать все числа
  - по модулю 2 (то есть сначала должны идти все чётные числа, а потом нечётные)
  - по модулю 3 (то есть сначала должны идти числа, делящиеся на 3, потом числа, которые дают остаток 1, при делении на 3, а потом числа, которые дают остаток 2)
  - по последней цифре
- Пусть есть вектор строк. Напишите функтор `SortByNthLetterFunctor`, который должен будет использоваться для сортировки строк по `n`-му символу. Например, следующий код должен будет отсортировать вектор строк по символу с индексом 2.

  ```
  vector<string> vs {"Cat", "Dog", "Axolotl", "Bear"};
  sort(vs.begin(), vs.end(), SortByNthLetterFunctor(2));
  ```

- В файле `3functor.cpp` есть пример, в котором есть функтор, который используется для хранения чисел, которые делятся на какое-либо число. Напишите аналогичный функтор, который принимает вектор строк и сохраняет в себе все строки, которые начинаются на определённую букву.
- В файле `movies.cpp` содержится заготовка кода. Отсортируйте массив `movies`, используя функторы:
  - по рейтингу
  - по названию
  - по дате

Часть 3: Стандартные функторы

Часть 4: Лямбда-функции

Ещё одним функциональным объектом является лямбда-функция:

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

int main() {
    std::vector v {18, 51, 2, 25, 14, 97, 73};
    std::sort(v.begin(), v.end(), [](int a, int b) {return a > b;});
    std::for_each(v.begin(), v.end(), [](int a) {std::cout << a << " ";});
}
```

- Используйте лямбда функцию и функцию `std::transform`, чтобы по вектору `v` создать вектор, содержащий последнии цифры чисел.
- В файле `movies.cpp` содержится заготовка кода. Отсортируйте массив `movies`, используя лямбда-функции:
  - по рейтингу
  - по названию
  - по дате
- Измените массив `movies`, с помощью `std::transform` и лямбда функций, так, чтобы
  - рейтинг каждого фильма уменьшился на 1
  - название каждого фильма было переведено в верхний регистр
- Создайте новый массив, который будет сорежать только фильмы с рейтингом `8` и выше. Используйте функцию `copy_if` и лямбда выражение.
- Удалите все фильмы из массива, который вышли в 90-е годы. Используйте функцию `remove_if` и `erase` и лямбда выражение.

Часть 5: Стандартные алгоритмы `STL`, принимающие функции

Часть 6: Лямбда-захваты

Часть 7: Тип-обёртка `std::function`

Часть 8: `std::bind`