Лямбда – Функции.

Функциональное программирование.

Функции для метода STL…

```
bool isPositive(int val) {
    return val > 0;
}

int main() {
    vector<int> v1 = { 1, 2, -1, -2, -3, 4 };
    if (all_of(v1.begin(), v1.end(), isPositive)) {
        cout << "POSITIVE" << endl;
    }
    else {
        cout << "NEGATIVE";
    }
    return 0;
}
```

Функции для STL методов писались за основной программой как самостоятельный модуль

Можно ли функцию “вшить” в метод STL?

То есть написать описание функции прямо в методе STL, чтобы она могла использоваться только методом

Да

Нет

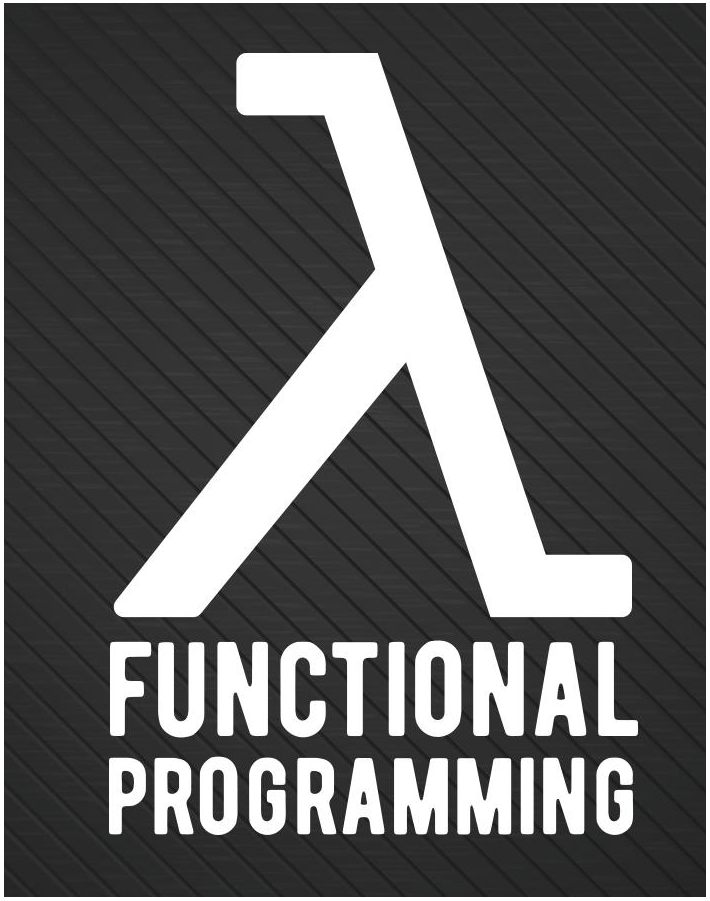
Если мы создадим отдельный объект

Лямбды – функции.
Функциональное программирование

Функциональное программирование

Функциональное программирование – стиль программирования, который опирается на вычисление математических функций, а не на выполнение команд.

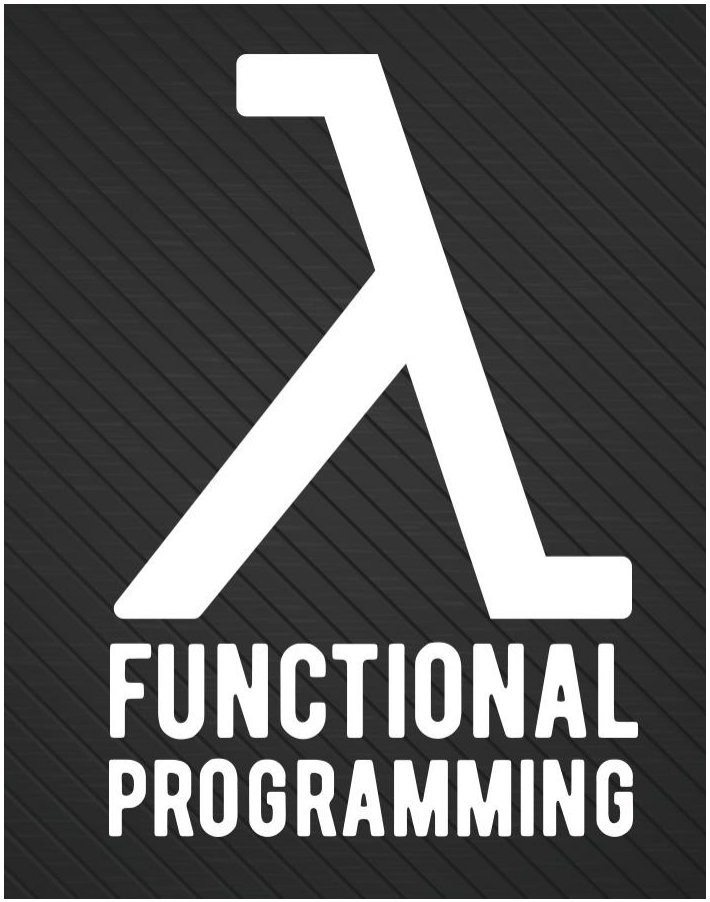
Функциональный язык: Haskell



Функциональное программирование

C++ взяло одно из достоинств функциональных языков – безымянные локальные функции, называемые *лямбда-функции.*

Их можно писать сразу внутри какого – либо выражения



Общий вид лямбда - функции

```
auto lambda_func(
   [] () {
   }
);
```

auto – чтобы определять автоматически тип

() – аргументы, которые передаются в лямбду функцию. Могут работать с элементами контейнера

[] - … вторая часть занятия ☺

Пример вывода значения

```
auto show_value(
   [](int _value) {
      cout << _value << endl;
   }
);

show_value(20);
```

Данное лямбда – выражение будет работать как обычная функция

Вшитая функция в метод STL

```
vector<int> v1 = { 1, 2, -1, -2, -3, 4 };
for_each(v1.begin(), v1.end(),
    [](int _val) {
        if (_val < 0) {
            _val = 0;
        }
    }
);
```

Для for_each прописана лямбда – функция, по которому он работает. Не нужно писать метод за Int main()

Вшитая функция в метод STL

```
vector<int> v1 = { 1, 2, -1, -2, -3, 4 };
for_each(v1.begin(), v1.end(),
    [](int _val) {
        if (_val < 0) {
            _val = 0;
        }
    }
);
```

Какие элементы будут лежать в векторе?

Практика! Пишем свои простые лямбда функции!

Может ли лямбда – функция работать с переменными, объявленные за область видимости лямбды?

✓ Да

✗ Нет

Да, если явно указать, с какой переменной работать

Список захвата

```
auto lambda_func(
   [] () {
   }
);
```

[ ] – список захвата. При назначении туда переменных, позволяет лямбда – функции работать с ними

Список захвата. Передача копий переменных

```
int some_value = 10;
int some_value2 = 20;
auto lambda_func(
   [some_value, some_value2]() {
      cout << some_value + some_value2;
   }
);
lambda_func();
```

Здесь переменные передаются как копии, то есть напрямую они редактироваться не будут.
Использование: границы диапазона для поиска элемента

Список захвата. Передача копий переменных

```
int some_value = 10;
int some_value2 = 20;
auto lambda_func(
   [some_value, some_value2]() {
      cout << some_value + some_value2;
   }
);
lambda_func();
```

Что будет выведено?

30

10

Список захвата. Передача по ссылке

```
int some_value = 10;
auto lambda_func(
   [&some_value]() {
      some_value += 100;
   }
);
cout << some_value;
```

Здесь переменная передается по ссылке, а значит ее можно редактировать напрямую.

Список захвата. Передача по ссылке

```
int some_value = 10;
auto lambda_func(
   [&some_value]() {
      some_value += 100;
   }
);
cout << some_value;
```

Чему будет равно some_value ?

10

110

Практика! Пишем лямбда функции со списком перехвата!