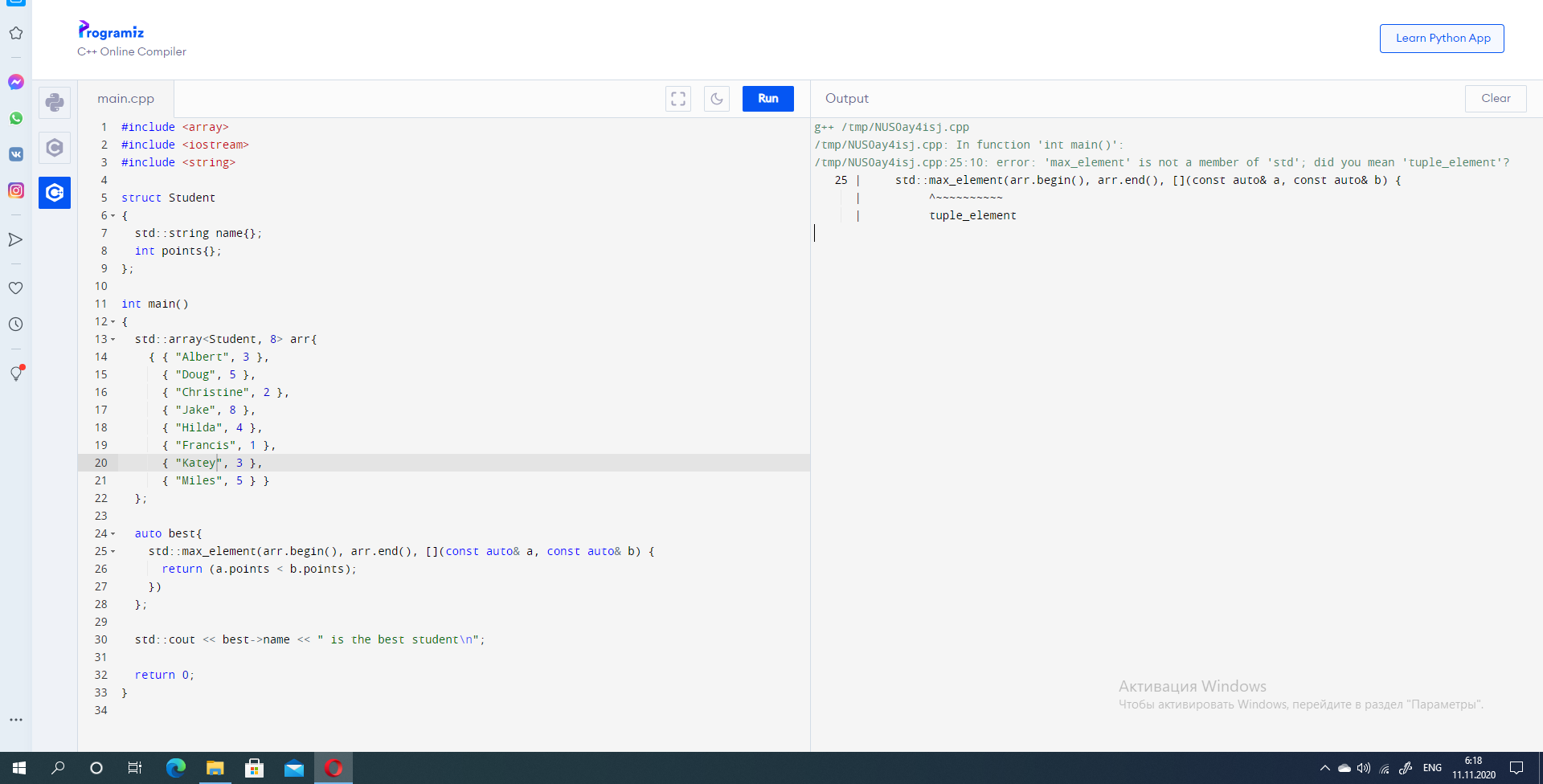
Лямбда функция - это особый вид функций, которые объявляются в месте использования и не получают уникального идентификатора для доступа к ним. Поддерживаются во многих языках программирования.

Обычно при создании анонимных функции - вызывают напрямую или ссылкой на функцию присваиваемой переменной, с помощью которой можно косвенно вызывать данную функцию. В последнем случае анонимная функция получает имя и уже перестаёт быть анонимной. Если функция ссылается на переменные, не содержащиеся в её теле, то функция называется замыканием.

Лямбда выражение - это типичная для многих языков синтаксическая конструкция для определения анонимной функции.

|  |  |
| --- | --- |
| ЯЗЫК | ПРИМЕР ЗАПИСИ |
| С++ | [](int x, int y){ **return** x + y; }  В С++ была добавлена возможность использовать лямбда-функции с auto:  **auto** lambda = [](**auto** x, **auto** y) {**return** x + y;};  Анонимная функция может захватывать как отдельные переменные, например:  int a; **auto** f = [a](){**return** a;}  так и все внешние переменные: по ссылке [&] или с помощью копии [=]. Также можно комбинировать эти подходы: например, все переменные захватить по ссылке, а определённые параметры по копии. Чтобы иметь возможность модифицировать переменные, захваченные по ссылке, нужно указать ключевое слово mutable при объявлении функции. В C++14 добавлена возможность инициализации переменных лямбды в захвате. Например:  [a = std::string{}](){**return** a;} |
| Delphi (версия 2009) | **function**(x, y: integer): integer  **begin**  result := x+y;  **end**; |
| JAVA (версия 8) | *// with no parameter*  () -> System.out.println("Hello, world.");  *// with a single parameter (This example is an identity function).*  a -> a  *//with a single expression*  (a, b) -> a + b  *// with explicit type information*  (Long id, String name) -> "id: " + id + ", name:" + name  *// with a code block*  (a, b) -> {**return** a + b;}  *// with multiple statements in the lambda body. It requires a code block.*  *// This example also includes a nested lambda expression as well as a closure.*  (id, newPrice) -> {  Optional<Product> mayBeProduct = findProduct(id);  mayBeProduct.ifPresent(product -> product.setPrice(newPrice));  **return** mayBeProduct.get();  } |
| JavaScript | (x, y) -> x + y |
| PHP | fn($x, $y) => $x + $y;  Стрелочные функции (Arrow Functions) добавлены в PHP 7.4.  *// PHP 5.3+*  **function**($x, $y) **use** ($a, &$b) { **return** $x + $y; }  Здесь $a, $b — захваченные переменные, при этом переменная $b ещё и замкнутая.  *// PHP 4.0.1+*  create\_function('$x, $y', 'return $x + $y;')  Создание анонимной функции с помощью функции create\_function. Такой способ объявлен устаревшим, начиная с PHP 7.2.0. |
| Python | **lambda** х, у: х+у |
| Pascal | (x, y) -> x + y |

Пример 1: C++



Пример 2: Python

