سلسلة تعلم البرمجة بلغة C++ الحديثة

Learn Modern C++ Programming Course

إعداد المهندس أحمد الديب

#18: Lambda Expressions

Function Pointers

```
template <class Iter, class T>
void operation(Iter first, Iter last, void (*op)(T&)) {
  for (auto it = first; it != last; it++) {
    op(*it);
  }
}

int main() {
  //
  std::vector<int> numbers{10, 20, 30, 40, 50};
  operation(numbers.begin(), numbers.end(), &increment<int>);
  operation(numbers.begin(), numbers.end(), &print<int>);
}
```

```
template <class T>
void increment(T& number) {
  number++;
}

template <class T>
void print(T& number) {
  std::cout << number << "\n";
}
```

Using Lambda

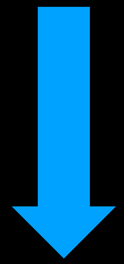
```
template <class T, class F>
void operation(T first, T last, F op) {
  for (auto it = first; it != last; it++) {
    op(*it);
  }
}

int main() {
  //
  std::vector<int> numbers{10, 20, 30, 40, 50};
  operation(numbers.begin(), numbers.end(), [](int& number) { number++; });

  operation(numbers.begin(), numbers.end(),
            [](int number) { std::cout << number << "\n"; });
}
```

Lambda Implementation

```
operation(numbers.begin(), numbers.end(), [](int& number) { number++; });
```



Check https://cppinsights.io

```
class increment {
  public:
   void operator()(int& number) const { number++; }
 };

 operation(numbers.begin(), numbers.end(), increment{});
```

Type of a Lambda

- The type of a lambda expression is implicitly defined to be the type of a function object of the equivalent class. So no two lambdas have the same type.

- When defining a lambda use `auto` or `std::function<R(AL)>` where R is the lambda's return type and AL is its argument list of types.

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

Lambda Parts

```
10
30
60
100
150
Min = 10 Max = 50 Sum = 0
```

```cpp
std::vector<int> numbers{10, 20, 30, 40, 50};

 int max = 0;
 int min = std::numeric_limits<int>::max();
 int sum = 0;

 auto minmaxsum = [&max, &min, sum](int number) mutable -> int {
   sum += number;
   if (number > max) max = number;
   if (number < min) min = number;
   std::cout << sum << "\n";
   return sum;
 };

 std::for_each(numbers.begin(), numbers.end(), minmaxsum);

 std::cout << "Min = " << min << " Max = " << max << " Sum = " << sum << "\n";
```

Capture List

Parameter List

Specifier

Return Type Declaration

Code to be executed

Thank you