## Lambda expression

- A function of no name.
- Also called anonymous function.
- Usually a parameter to a higher order function, or on the right side of an auto declaration.

# Lamba expression

- The simplest possible expression of lambda in C++ is [](){}.
- Three parts:
  - [ ]: capture list (more later)
  - (): where we specify parameters
  - {}: code

# Example of lambda expression

```
template <class FUNC>
void do_n_times(FUNC fn, int n) {
   for(int i=0; i<n; ++i) fn(); }
template <class FUNC, class T>
void applyfn(FUNC fn, T fnparam) {
  fn(fnparam); }
int main() {
  // This function only exists at current scope.
  auto respond = [ ] (const char * x) { cout << x << " isn't here." << endl; };
  // calling a function that has no name...
  applyfn([ ] (int x) { cout << "Here's my number: " << x << endl; }, 8675309);
  // "Call me maybe" printed for five lines.
  do_n_times( [ ] () { cout << "Call me maybe." << endl; }, 5 );
  // lambdas can be called the same way as functions
  respond("Jenny"); }
```

## capture

- Capture is a list of variables found in the scope outside the lambda that are to be made available inside the lambda expression.
- Can capture by value or by reference.
- Capture by value: [var1, var2, var3]
- Capture by reference: [&var1, &var2]
- The behavior of these within the lambda will be as one would expect if the captured variables had been passed as parameters, but whatever calls the lambda still only need to provide the parameters specified in the () portion.

## Capture by Reference

```
template <class FUNC>
void do_n_times(FUNC fn, int n) {
for(int i=0; i<n; ++i) fn(); }
int main() {
  int i=1;
  do_n_times( [i] () {
      cout << i; /* can't do i++ here, why not? */ }, 3 );
  do_n_times( [&i] () {
     cout << i++; }, 3 );
}
// outputs 111123
```

# Example as function pointer in higher order function :

*Before:*

```
bool greater10(int value) { return (value > 10); }
int main() {
   …
   auto loc = find_if(v.begin(), v.end(), greater10);
    …
}
```

Now:

```
auto loc = find_if(v.begin(), v.end(), [] (int value) { return (value > 10);});
```

## Exercise

- Use lambda expression to find if a string (str) has alpha-numeric characters (isalnum(char)).

## Solution

```
auto first = find_if(str.begin(), str.end(), [ ](char c) { return isalnum(c); });

int num = count_if(str.begin(), str.end(), [ ](char c) { return c == ‘a’; });
```