Ch 8: Statements

CSCI 330

Overview

- **Statements** are building blocks of control flow and program structure
- Categories:
  - Expression statements
  - Declarations
  - Control flow: selection, iteration, jumps
  - Scoping: compound, functions, namespaces
- Modern enhancements: constexpr if, structured bindings, attributes

Expression and Compound Statements

- Expression statements: end with ;

```
x = 5
std::cout << x;
```

- Compound statement: block of code in {}

```
{
    int x = 0;
    x++;
}
```

Declaration Statements

- Introduce and define variables or types
  - int a = 42;
  - const double pi = 3.14;
- auto, constexpr, extern, static, using all apply to declarations

Functions and Namespaces

- Function definition:

```
int add(int a, intb) {
  return a+b;
}
```

- Namespace defines a scope:

```
namespace math {
   int square(int x) {return x * x; }
}
```

using Declaration and Type Aliases

- using Directive:

  using namespace std;

- using Declaration:

  using std:: string;

- Type alias:

  using IntVec = std::vector<int>;

Structured Bindings

- Decompose tuples, pairs, structs

```
auto [x, y] = std::make_pair(1,2);
```

- works with arrays, custom types with public members

Attributes

- Compiler hits that don’t change semantics

```
[[nodiscard]] int compute();
[[maybe_unused}} int x;
```

- portable, standard way to express intent

Selection Statements: if and constexpr if

- if / else statement (conditional test, branch: true, false)

  if (x> 0) {true } else { false}

- if constexpr (c++17): (conditional compilation logic) enables compile-time branching, no code generated for false branch

  if constexpr (std::is_integral_v<T> { . . . }

- Switch: case, labels, default, and fallthrough

```
switch (value) {
   case 1: . . . .; break;
   default: . . . . .;
}
```

Iteration Statements

- while / do
- for loop (classic)
- Range-based for
- Range Expression)

```
//while / do while
while (i < 10 ) { ++i; }
```

```
do {
    ++i;
} while (i < 10);
```

```
//classic for loop
for (int i = 0; i < 10; ++i) {
    std::cout << i;
}
```

```
//range based for loop
for (int x : vec) { std::cout << x;}
```

```
for (auto& c: "hello"s) { . . . }
```

Jump statements

- break / continue
  - break: exit loop/switch
  - continue: skip rest of loop iteration
- goto
  - use discouraged
  - jumps to label
- return
  - Ends a function and returns control (and optional value)