Outline

- In this lesson, we will:
  - Show how to initialize local variables
  - See how to assign a local variable a new value

Background

- In the previous topic:
  - We declared a local variable
    ```cpp
    int n;
    char ch;
    std::string name;
    double x;
    bool is_valid;
    ```
  - The local variable was then given a value by executing:
    ```cpp
    std::cin >> variable_name;
    ```

Initialization

- What happens if simply declare an identifier to be a local variable?

```cpp
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    int n;
    char ch;
    double x;
    bool is_valid;
    std::cout << n        << std::endl;
    std::cout << ch        << std::endl;
    std::cout << x        << std::endl;
    std::cout << is_valid << std::endl;
    return 0;
}
```

Output:

```
32765
S
5.17236e-159
133
```
Initialization

- The local variables are stored in main memory
  - Whatever 0s and 1s are currently there are those interpreted as either an integer, a character, a float, or a Boolean value

```
#include <iostream>
// Function declarations
int main();
// Function definitions
int main() {
    int n{0};
    char ch{'};   // The null character
    double x{0.0};
    bool is_valid{false};
    std::cout << n << std::endl;
    std::cout << ch << std::endl;
    std::cout << x << std::endl;
    std::cout << is_valid << std::endl;
    return 0;
}
```

Output:
```
0
'
0
```
Initialization

• Programming principle:
  In any general application, all variables must be initialized,
  either with their default value or a value you choose.
In an embedded system, a variable may be left uninitialized,
  but only with an appropriate comment explaining why.

```cpp
int register_number;    // Uninitialized
// - will be assigned during an interrupt
```

Assignment

• A local variable, once declared and initialized may have a new value
  assigned to it:

```cpp
#include <iostream>

// Function declarations
int main();

// Function definitions
int main() {
    // Local variable declaration
    int n = 42;
    std::cout << "The value of 'n' is " << n << std::endl;
    n = 91;
    std::cout << "The value of 'n' is " << n << std::endl;
    n = 1970;
    std::cout << "The value of 'n' is " << n << std::endl;
    return 0;
}
```

Integers

• On occasion, you may see

```cpp
int m{};
int n{};
```

```cpp
m = n = 10;
```

• This is the same as:

```cpp
n = 10;
m = 10;
```
Summary

- Following this lesson, you now:
  - Understand the need for initializing local variables
  - Know the default initial values
  - Know how to assign a local variable a new value
  - The = operator is the assignment operator

References


Acknowledgements

None so far.

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using Consolas.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see https://www.rbg.ca/ for more information.
Disclaimer

These slides are provided for the ECE 150 Fundamentals of Programming course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.