Keywords

- Special class of identifiers
- C++ reserved words
- Cannot be used as names for variables or anything else

Examples:

```
bool  char  default  false  new  this
break  class  else  if  return  true
```

Variables

- Memory location to store data for a program
- Start with a letter or underscore and rest must be letters, digits or underscores
- Case sensitive
- Use meaningful names
- Use mixed case
  - e.g. `timeOfArrival`

Chapter 1

C++ Basics

Hello World

```
// File: helloWorld.C
#include <iostream>    // Library for input and output
using namespace std;  // Location of iostream library
int main() {          // Execution starts here
    cout << "Hello World!" << endl;
    return 0;          // Execution ends here
}
```

- To compile: `g++ helloWorld.C`
- To run: `./a.out`
Assignment Statements

- Used to change value of a variable
- Equal sign (=) is assignment operator
- Variable on left side, expression on right
  - salary = rate * hours;
- Type mismatch
  - int x = 2.99; // some compilers will complain
  - double y = 2; // this is ok
  - bool flag = 0; // false - 0, true - 1

Variables

- Which of these are legal names?
  - _abc
  - _myRATE_
  - %change
  - 3X
  - file.name
  - data-1
  - amtOf$
  - ABC123z7
  - 12

Literals and Constants

- Literals cannot change values during execution
  - 2, 5.75, 'Z', “hello world”
- Use named constants instead (when possible)
  - const int BRANCH_COUNT = 100;
- Common naming convention for const

Variables

- Must be declared before use
  - One per line or combined
  - At beginning of block or just before first use
- Must be initialized

Simple types:

- short int
- int
- long int
- float
- double
- char
- bool
- long double
Type Casting

- Used to change a value of one type to a value of another type
- Does not change the original variable

```
int num = 1, denom = 2;
double result;

result = static_cast<double>(num) / denom;
```

Arithmetic Operators & Expressions

- Operators: + - * / %
- Standard precedence rules apply
  - Use parentheses to reduce errors and increase readability
  - \( x + y * z \) should be \( x + (y * z) \)
- Assignment shorthands available

```
<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>count += 2;</td>
<td>count = count + 2;</td>
</tr>
<tr>
<td>total -= discount;</td>
<td>total = total – discount;</td>
</tr>
<tr>
<td>amt *= count1 + count2;</td>
<td>amt = amt * (count1 + count2);</td>
</tr>
</tbody>
</table>
```

Increment & Decrement

- Shorthand to add or subtract 1 from a variable: \( \text{m}++ \), \( \text{n}-- \)
- Pre-increment: \( ++\text{m} \)
  - Increments variable then uses new value
- Post-increment: \( \text{m}++ \)
  - Uses current value of variable then increments it

```
int n = 3;
int ans = 2 * (n++);
ans equals ?
```

Arithmetic Precision

- Highest order operand determines precision
  - If all values are int, result is int
  - If one value is double/float, result is double/float
  - Division with negative numbers varies

```
10/4 equals ?  10/4.0 equals ?
int num = 1, denom = 2;
num / denom equals ?
1 / 2 / 3.0 / 4 equals ?
```
**Formatting Numbers**

- To display numbers with fixed digits after the decimal place use these flags

```cpp
cout.setf(ios::fixed);
cout.setf(ios::showpoint);

cout.precision(2);
cout << “The price is $” << price << endl;

cout.precision(3);
cout << “The extended price is $” << price << endl
```

---

**Cin**

- Always include a prompt before every cin
- Can read in multiple values separated by a space
- Datatype matters

```cpp
int hours, mins;
char colon;
cout << “Enter the time in 24hr format (hh:mm) : “;
cin >> hours >> colon >> mins;

int age, height;
cout << “Enter your age followed by your height in inches: “;
cin >> age >> height;
```

---

**Console Input/Output**

- cout for output, cin for input
  - cout << “Hello World”;
  - cin >> name;
- Must include `<iostream>` library
- Must use std namespace

```cpp
#include <iostream>
using namespace std;

// Any program which contains input or output should start with these lines

//
```

---

**Cout**

- Can output variables, constants, literals & expressions
- Use \n or \endl for a newline
- Use \t for a tab
- Separate variables from strings using <<

```cpp
cout << “The cost of the “ << movieName
  << “ DVD is $” << (basePrice + tax) << endl;
```

---
General Program Style

• Code must be easy to read and modify
  – Must have adequate whitespace
• Use lots of comments
  – // for one line comments
  – /* … */ for multi line comments
• Variable names must be meaningful

CS1129 Style Guide

• Indent by 2 spaces
• Reserve /* */ for debugging
• Leave a space after //
• Declare all variables at top of block
• Use Java style braces
• Variable naming
  – ALL_CAPS for constants
  – lowerToUpper for variables