CMPE 180-92

Data Structures and Algorithms in C++

August 24 Class Meeting

Department of Computer Engineering
San Jose State University

Fall 2017
Instructor: Ron Mak
www.cs.sjsu.edu/~mak

Basic Info

- Office hours
  - TuTh 3:00 – 4:00 PM
  - ENG 250
- Website
  - Faculty webpage: http://www.cs.sjsu.edu/~mak/
  - Class webpage: http://www.cs.sjsu.edu/~mak/CMPE180-92/
  - Syllabus
  - Assignments
  - Lecture notes

Permission Codes?

- If you need a permission code to enroll in this class, see the department's instructions at https://cmpe.sjsu.edu/content/Undergraduate-Permission-Number-Requests
- Complete the Google form at https://docs.google.com/a/sjsu.edu/forms/d/e/1FAIpQLSe9YgAea-QsgLZof-KIMmuQthoChL4micudyRukgWneiByN2A/viewform

Course Objectives

- The primary goal of this class is to learn a useful subset of C++ programming language and fundamental data structures and algorithms expressed in C++.
- You will learn best practices for developing software.
- You will acquire software development skills that are valued by employers.

Course Objectives, *cont'd*

- Not course objectives:
  - Complete knowledge of C++
    - We will briefly touch the new features of C++ 11 and 14.
  - Advanced data structures and algorithms
  - Advanced algorithm analysis

C++ Tutoring

- We hope to provide C++ tutoring during the week by the instructional student assistants (ISAs).
  - Students have found this very helpful.
  - Please take advantage of this service!
- ISAs and their schedules to be announced.

Required Textbooks

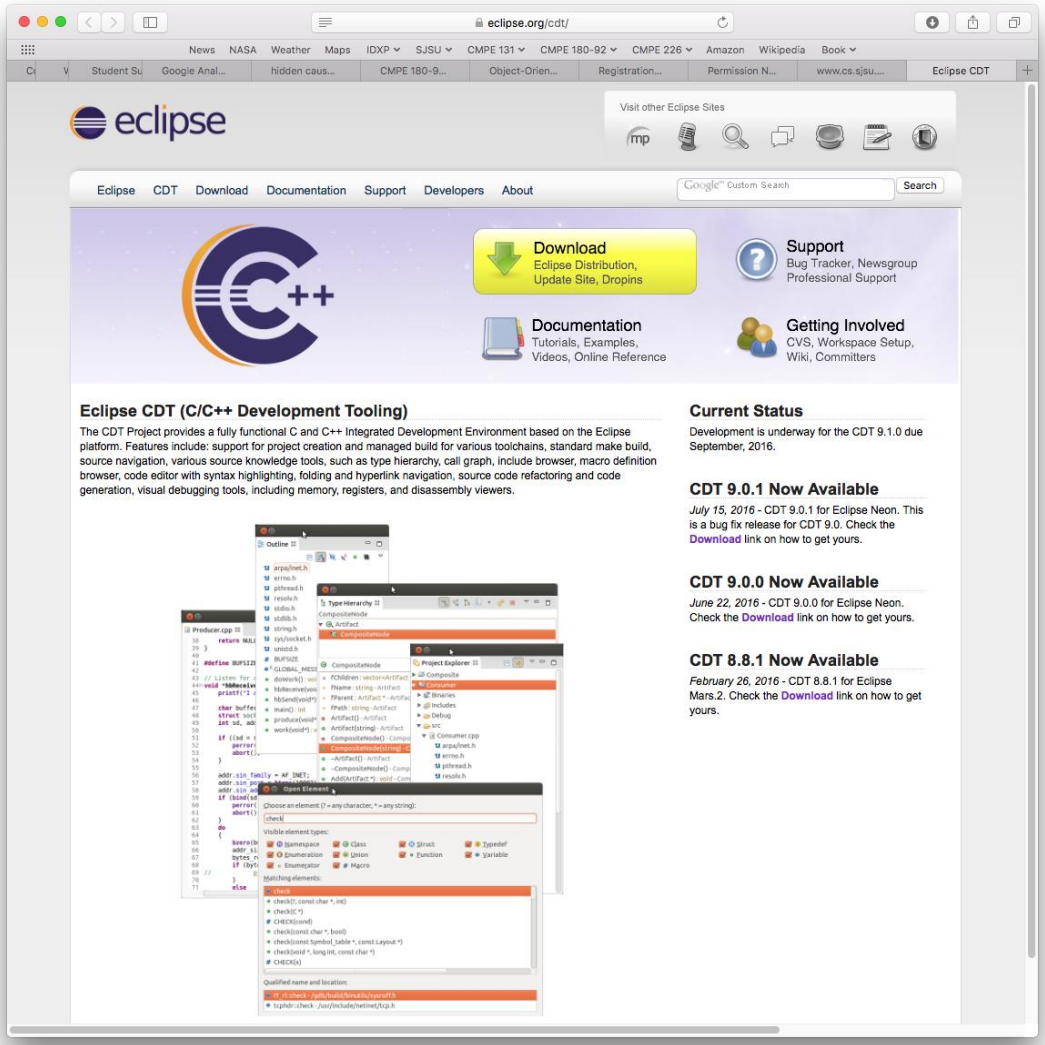
- **Problem Solving with C++**, 10th edition
  - Author: Walter Savitch
  - Publisher: Pearson, 2017
  - ISBN: 978-0134448282
- **Data Structures Using C++**, 2nd edition
  - Author: D.S. Malik
  - Publisher: Cengage Learning, 2010
  - ISBN: 978-0324782011

You are responsible for doing the chapter readings before each class, as indicated in the class schedule. In-class quizzes will be based on the readings.

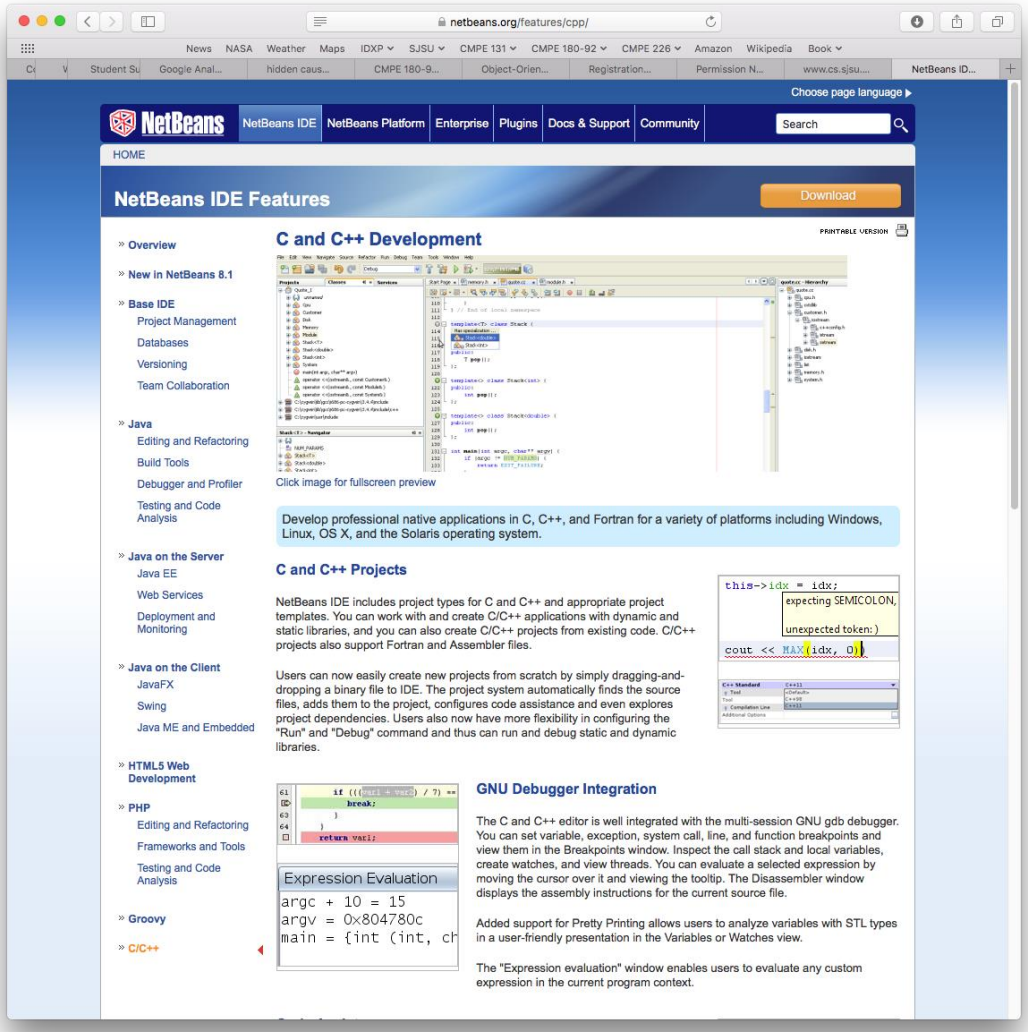
Software to Install

- Install one of the following integrated development environments (IDE) for C++ development on the Mac or Linux platform:
  - Eclipse CDT (C/C++ Development Tooling): https://eclipse.org/cdt/
  - NetBeans C and C++ Development: https://netbeans.org/features/cpp/

Software to Install, *cont'd*



Software to Install, *cont'd*



C++ on the Mac and Linux Platforms

- GNU C++ is usually pre-installed on the Mac and Linux platforms.
- No further action required!
- Avoid using Apple's Xcode on the Mac for this class.
  - You run the risk of writing programs that will not port to other platforms.

C++ on Windows

- The Windows platform has proven to be problematic for this class.
  - Difficult to install the Cygwin environment correctly.
  - Difficult to install C++ libraries successfully.
- Avoid using Microsoft's Visual C++ on Windows for this class.
  - You run the risk of writing programs that will not port to other platforms.

C++ on Windows, *cont'd*

- Run Linux in a virtual machine on Windows.
- Use Linux's pre-installed GNU C++ environment.

We will not provide support for Windows.

If you insist on running Windows,
you are on your own!

C++ on Windows, *cont'd*

- Steps:

1. Download and install the VirtualBox virtualizer: https://www.virtualbox.org/wiki/VirtualBox
2. Start VirtualBox.
3. Download a Linux .iso image (such as Debian, https://www.debian.org) and install it inside VirtualBox.
4. Start Linux from inside VirtualBox.
5. Download and install Eclipse or NetBeans on Linux.

More detailed VirtualBox instructions to come.

C++ 2011 Standard

- We will use the 2011 standard of C++.
- You must set this standard explicitly for your project in Eclipse and in NetBeans.
- On the command line:

```
g++ foo.cpp -std=c++11 -o foo
```

Set the C++ 2011 Standard in Eclipse

- Right-click on your project in the project list at the left side of the window.
- Select “Properties” from the drop-down context menu.
- In the left side of the properties window, select “C/C++ Build” ➔ “Settings”.
- In the Settings dialog, select “GCC C++ Compiler” ➔ “Dialect”.
- For “Language standard” select “ISO C++ 11”.
- Click the “Apply” button, answer “Yes”, and then click the “OK” button.

Set the C++ 2011 Standard in NetBeans

- Right-click on your project in the project list at the left side of the window.
- Select “Properties” from the drop-down context menu.
- In the left side of the properties window, select “Build” ➔ “C++ Compiler”.
- In the table, for “C++ Standard” select “C++11”.
- Click the “Apply” button and then click the “OK” button.

Assignments

- You will get lots of programming practice!
  - Multiple programming assignments per week.
  - Several small practice problems that emphasize specific skill needed to solve the main assignment.
- We will use the online CodeCheck system which will automatically check your output against a master.
  - You will be provided the URL for each assignment.
  - You can submit as many times as necessary to get the correct output.

Assignments, *cont'd*

- Assignments will be due the following week, before the next lecture.
- Solutions will be discussed at the next lecture.
- Assignments will not be accepted after solutions have been discussed in class.
  - Late assignments will receive a 0 score.

Individual Work

- You may study together.
- You may discuss the assignments together.
- But whatever you turn in must be your individual work.

Academic Integrity

- Copying another student's work or sharing your work is a violation of academic integrity.
- Violations will result in harsh penalties by the university.
  - Academic probation.
  - Disqualified for TA positions in the university.
  - Lose internship and OPT sponsorship at local companies.
- Instructors must report violations.

Moss

- Department policy is for programming assignments to be run through Stanford University's Moss application.
  - Measure of software similarity
  - Detects plagiarism
  - http://theory.stanford.edu/~aiken/moss/
- Moss is not fooled by
  - Renaming variables and functions
  - Reformatting code
  - Re-ordering functions

Example Moss output:
http://www.cs.sjsu.edu/~mak/Moss/

Quizzes

- In-class quizzes check your understanding of:
  - the required readings
  - the lectures
- Quizzes will be conducted online using Canvas.
  - Each quiz will be open for only a very short time period, around 15 minutes.
  - You are responsible for bringing a laptop or mobile device to class that can connect to the wireless.
- There will be no make-up quizzes.

Exams

- The quizzes, midterm, and final examinations will be closed book.
- Instant messaging, e-mails, texting, tweeting, file sharing, or any other forms of communication with anyone else during the exams violates academic integrity.

Exams, *cont'd*

- There can be no make-up midterm examination unless there is a documented medical emergency.
- Make-up final examinations are available only under conditions dictated by University regulations.

Final Class Grade

- 50% assignments
- 15% quizzes
- 15% midterm
- 20% final exam

- The class is graded CR/NC.

- Students who have a weighted score above the passing threshold at the end of the semester will receive the CR grade.
  - We expect least 75% of students will pass.

Fast Pace!

- This class will move forward at a fast pace.
- Lectures will consist of:
  - New PowerPoint slides by the instructor
  - PowerPoint slides from the textbook publishers
  - Program examples and live demos
  - In-class quizzes
  - Questions, answers, and discussion
- Lecture materials will be posted to the class webpage: http://www.cs.sjsu.edu/~mak/CMPE180-92/index.html

Piazza

- Besides Canvas, we will use Piazza.
  - Announcements
  - Online forum for discussions about the class
- You will receive an email invitation to join Piazza.
  - Sent to the email address that the university has on record for you.

What is C++

- An object-oriented programming (OOP) language.
  - Supports encapsulation, inheritance, polymorphism.
  - Based on the C language with added OOP features.
- A complex language!
  - Lots of features.
  - Somewhat arcane syntax.
  - Easy to make programming errors.
  - Things happen automatically at run time | that you may not expect.

A Useful Subset of C++

- We will only learn a useful subset of C++.
  - Very few people (not including your instructor) know all of the language.
  - Among professional C++ programmers, everybody knows a different subset, depending on experience, training, and application domains.
- It will be easy to stumble accidentally into an obscure language feature.
  - We'll have to figure out together what happened!

Our First C++ Program

- The infamous “Hello, world!” program.

helloworld.cpp

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

int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

```
~mak/CMPE180-92/programs: g++ helloworld.cpp -o helloworld
~mak/CMPE180-92/programs: ./helloworld
Hello, world!
```

Algorithms and Program Design

Savitch_ch_01.ppt: slides 57– 60

- Display 1.4
  - Compiling and Running a C++ Program
- Display 1.5
  - Preparing a C++ Program for Running
- Display 1.7
  - Program Design Process

Sample Program 1-8: Pods and Peas

Savitch_ch_01.ppt: slides 34 – 44

- Sample program 1.8

Break

Identifiers, Variables, and Keywords

- Identifiers are names.
- Variables represent values that can change.
  - Variables have names (variable identifiers).
  - Declare variables before you use them.
    - A declaration tells what is the variable's type (integer, real, character, etc.).
    - A declaration can also give an initial value to the variable.
- Keywords are reserved by C++ and cannot be used as identifiers.
  - Examples: `if for while`

Assignment Statements

- At run time, be sure to initialize a variable (give it a value) before you use it.
  - Either initialize the variable when you declare it.
    - Example: `int i = 5;`
  - Or execute an assignment statement.
    - Example: `i = 10;`
- Do not confuse = (assignment) with == (equality comparison).

```
i = 10;         // assign the value of 10 to variable i
if (i == 10)    // test whether or not i is equal to 10
```

Input and Output

- Input stream
  - Data read by the program at run time.
  - Standard input stream: `cin` (default: the keyboard).
  - Example: `cin >> x >> y;`

    extraction operator
    - Extract (read) the next two values from the keyboard and assign them to `x` and `y`, respectively.

- Output stream
  - Written by the program at run time.
  - Standard output stream: `cout` (default: the display).
  - Example: `cout << "x equals " << x << endl;`
    - Insert (write) to the display.

insertion operator

`#include` and `using namespace`

- `#include <iostream>`
  - Read in the definitions of `cin` and `cout`.
- `using namespace std;`
  - Make the names `cin` and `cout` that reside in the standard namespace `std` available to the program.
  - Many other names reside in the standard namespace.

Formatting Real Numbers for Output

- Call methods of `cout` to format real numbers.
- `cout.setf(ios::fixed);`
  - Use fixed-point notation (not scientific).
- `cout.setf(ios::showpoint);`
  - Always show the decimal point.
- `cout.precision(2);`
  - How many decimal places.

Input From `cin`

- `cin >> v1 >> v2 >> v3;`
  - Read values into multiple variables.
  - The input values should be separated by spaces.
- The values are not read until you press the return key.
  - Therefore, you can backspace and make corrections.

Some Basic Data Types

- A data type determines
  - what kind of data values
  - what operations are allowed
- Data type **`int`** for integer values without decimal points.
  - Examples: **`0 2 45 -64`**
- Data type **`short`** for small integer values.
- Data type **`long`** for very large integer values.

Some Basic Data Types, *cont'd*

- Data type **`double`** for real numbers.
  - Fixed-point notation: **`34.1 23.0034 -1.0 89.9`**
  - Scientific notation: **`3.67e17 5.89E-6 -7.23e+12`**
- Data type **`float`** for less precision and smaller magnitude.
- Data type **`char`** for individual characters.
  - Examples: **`'a'  'Z'`**
  - Use only single quotes for character constants in a program.

Some Basic Data Types, *cont'd*

- Data type **`bool`** for the Boolean values **`true`** and **`false`**.
- The Boolean value **`false`** is stored as the integer 0.
- The Boolean value **`true`** is stored as the integer 1.

`cin` Skips Input Blanks

- The statements

```
char ch1, ch2;
cin >> ch1 >> ch2;
```

when given the input

```
A  B
```

will set `ch1` to `'A'` and `ch2` to `'B'`.

> `cin` uses blanks and line feeds to separate input data values, but otherwise it skips the blanks and line feeds.

String Type

- `#include <string>`
  - Required if your program uses strings.
- Enclose string values with double quotes in your program.
  - Example: `"Hello, world!"`
- To input a string from `cin` that includes spaces, all in one line:

```
string str;
getline(cin, str);
```

Type Compatibilities and Conversions

- `int pi = 3.14;`
  - `double` → `int` is invalid. You cannot set a `double` value into an `int` variable .
- Some valid conversions:
  - `int` → `double`
  - `char` → `int`
  - `int` → `char`
  - `bool` → `int`
  - `int` → `bool`

Any nonzero integer value is stored as true. Zero is stored as false.

Arithmetic

- Arithmetic operators: `+ - * / %`
- Integer `/` result if both operands are integer.
  - Quotient only.
- Use the modulo operator `%` to get a remainder.
- Double `/` result (includes fractional part) if either or both operands are double.

Operator Shorthand

- `n += 5` shorthand for `n = n + 5`
- `n -= 5` shorthand for `n = n - 5`
- `n *= 5` shorthand for `n = n*5`
- `n /= 5` shorthand for `n = n/5`
- `n %= 5` shorthand for `n = n%5`

The `if` Statement

- Example `if` statement:

```
if (n <= 0)
{
    cout << "Please enter a positive number." << endl;
}
```

- Example `if else` statement:

```
if (hours > 40)
{
    gross_pay = rate*40 + 1.5*rate*(hours - 40);
}
else
{
    gross_pay = rate*hours;
}
```

`while` Loops

- Example `while` loop:

```
while (count_down > 0)
{
    cout << "Hello ";
    count_down = count_down - 1;
}
```

- Example `do while` loop:

```
do
{
    cout << "Hello ";
    count_down = count_down - 1;
} while (count_down > 0)
```

Named Constants

- It's good programming practice to give names to constants:

```
const double PI = 3.1415626;
```

- Easier for humans to read the program.
- Easier to modify the program.
- Convention: Use ALL_CAPS for the names of constants.

Boolean Operators

- Relational operators: `==` `!=` `<` `<=` `>` `>=`
- And: `&&`
- Or: `||`
- Not: `!`
- Short-circuit operation: `p && q`
  - `q` is not evaluated if `p` is false
- Short-circuit operation: `p || q`
  - `q` is not evaluated if `p` is true

Precedence Rules

Savitch_ch_02.ppt: slide 101

Precedence Rules

The unary operators +, –, ++, --, and !.
The binary arithmetic operations *, /, %
The binary arithmetic operations +, –
The Boolean operations <, >, <=, >=
The Boolean operations ==, !=
The Boolean operations &&
The Boolean operations ||

Highest precedence (done first)

Lowest precedence (done last)

Enumeration Types

- A data type with values defined by a list of constants of type **`int`**
  - Examples:

```
enum Direction {NORTH, SOUTH, EAST, WEST};

enum MonthLength{JAN_LENGTH = 31,
                 FEB_LENGTH = 28,
                 MAR_LENGTH = 31,
                    …
                 DEC_LENGTH = 31};
```

Nested `if` Statements

- Example:

```
if (net_income <= 15000)
{
    tax_bill = 0;
}
else if ((net_income > 15000) && (net_income <= 25000))
{
    tax_bill = (0.05*(net_income - 15000));
}
else // net_income > $25,000
{
    five_percent_tax = 0.05*10000;
    ten_percent_tax = 0.10*(net_income - 25000);
    tax_bill = (five_percent_tax + ten_percent_tax);
}
```

The `switch` Statement

- Use a **`switch`** statement instead of nested **`if`** statements to compare a single integer value for equality.
  - Note the need for the **`break`** statements.
  - Note the **`default`** case at the bottom.

```
int digit;
...
switch(digit)
{
   case 1: digit_name = "one";   break;
   case 2: digit_name = "two";   break;
   case 3: digit_name = "three"; break;
   case 4: digit_name = "four";  break;
   case 5: digit_name = "five";  break;
   case 6: digit_name = "six";   break;
   case 7: digit_name = "seven"; break;
   case 8: digit_name = "eight"; break;
   case 9: digit_name = "nine";  break;

   default: digit_name = ""; break;
}
```

The Increment and Decrement Operators

- `++n`
  - Increase the value of `n` by 1.
  - Use the increased value.
- `n++`
  - Increase the value of `n` by 1.
  - Use the value before the increase.

The Increment and Decrement Operators, *cont'd*

- `--n`
  - Decrease the value of `n` by 1.
  - Use the decreased value.
- `n--`
  - Decrease the value of `n` by 1.
  - Use the value before the decrease.

`for` Loops

- Example:

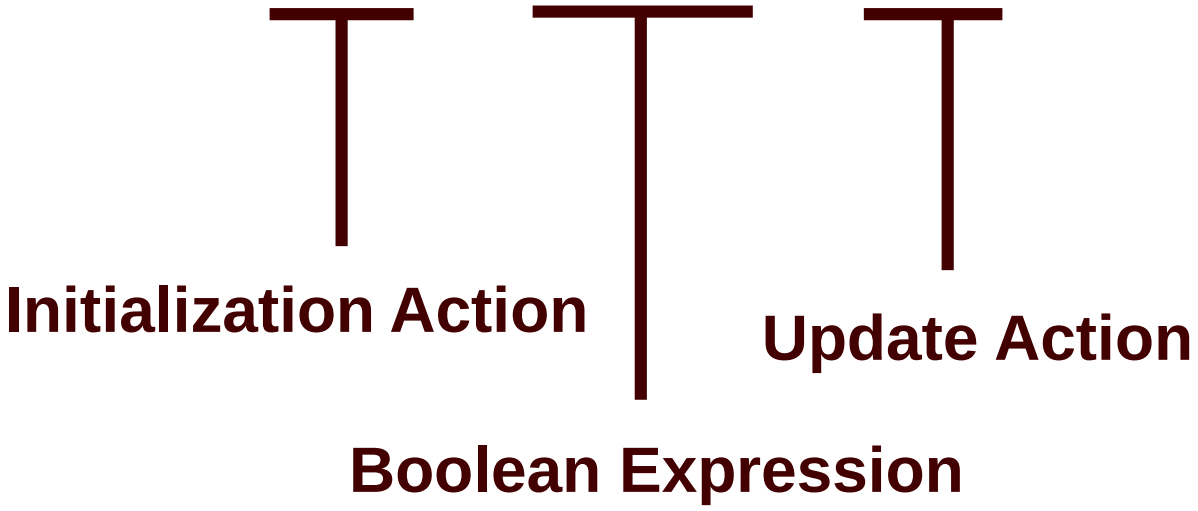
```
int sum = 0;

for (int n = 1; n <= 10; n++)
{
    sum = sum + n;
}

cout << "The sum of the numbers 1 to 10 is "
     << sum << endl;
```

Note that variable `n` is local to the loop body.

`for` Loops, *cont'd*

- The **`for`** loop uses the same components as the **`while`** loop, but in a more compact form.

```
for (n = 1; n <= 10; n++)
```



The `break` Statement

- Use the **`break`** statement to exit a loop before "normal" termination.
- Do not overuse!
  - Well-designed loops should end normally.
- This use of **`break`** is different from the necessary use of **`break`** in a **`switch`** statement.

Loop Considerations

- Choosing the right kind of loop to use
- Designing loops
- How to control a loop
- How to exit from a loop
- Nested loops
- Debugging loops

Practice Problems for Week 1

- Five problems in CodeCheck:
  - https://play.codecheck.ws/files?repo=fall2017&problem=w1-1
  - https://play.codecheck.ws/files?repo=fall2017&problem=w1-2
  - https://play.codecheck.ws/files?repo=fall2017&problem=w1-3
  - https://play.codecheck.ws/files?repo=fall2017&problem=w1-4
  - https://play.codecheck.ws/files?repo=fall2017&problem=w1-5
- Submit as many times as you need to get correct results for each problem.
  - No penalties for multiple submissions.
- Download all five signed zip files and submit them together into Canvas: **Week 1 Practice**
  - Do not rename the zip files.

Main Assignment for Week 1

- Assignment #1 will give you practice with C++ control statements and nested loops.
  - Write-up:
  - Input file:
  - CodeCheck URL: http://codecheck.it/codecheck/files/17082218369vfzb082gwl1ggr02jbdp0cn6
- Follow carefully the instructions on how to use CodeCheck and how to submit into Canvas.