

CSC 221: Computer Programming I

Fall 2001

- C++ basics
 - program structure: comments, #include, main, return
 - output: streams, cout, endl
 - input: variables, identifiers
 - expressions: data types, operators, strings
- using Visual C++
 - projects, C++ source files, build & run

syntax & semantics

syntax: the precise form that programs & instructions must follow

- computers are very picky
- at first, we will spend time learning the computer's rules
(e.g., semi-colon here, no space there)
- don't get the impression that syntax is what programming is about
foreign language analogy: vocabulary first → literature

semantics: the meaning of the programs & instructions

- computers are literal-minded
i.e., they do what you say, not what you mean!
- we will develop a repertoire of instructions/tools for solving problems
carpentry analogy: simple tools first → artisanship
- don't get the impression that programs are trivial, silly things
we will start simple and build complexity as we learn

first C++ program

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

when compiled & executed, this program writes the message

Hello and welcome to CSC221.

on the screen

program components: comments

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

any text that appears on a line preceded by // is a comment

- comments are completely ignored by the C++ compiler, so why have them?
- can also specify comments using notation: /* comment */

every program you write should have a comment block at the top with

➤ file name, author name, date, short description

program components: main

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

every program must have a function named main

- this is the control center of the program
whatever statements appear inside the curly braces are executed in order when the program executes
- the function must "return" an integer value
don't worry about it for now, just put "return 0;" at end

in general:

```
int main()
{
    statement1;
    statement2;
    . . .
    statementN;

    return 0;
}
```

program components: #include

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

C++ provides many useful *libraries* (separate files of code)

- libraries are loaded using the #include directive
`#include <iostream>` loads the `iostream` library, containing input/output routines
- note: in-class examples will follow the ANSI/ISO standard
library files do not end in `.h`, must include `using namespace std;` at end

program components: output statements

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

`cout` is the standard C++ output stream, used for writing messages
stream analogy: place message on stream, current carries it out

- the stream insertion operator `<<` is used to place messages on the stream
text placed inside quotes is displayed as is (without the quotes)
`endl` is a special keyword symbolizing a new-line character
can place multiple items on the stream using multiple instances of `<<`

program components: output statements (cont.)

```
// hello2.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl << endl;

    cout << "Programming is both intellectually challenging" << endl
         << "and artistically rewarding. Enjoy!" << endl;
    cout << "                -- Dr. Reed" << endl;

    return 0;
}
```

- can have multiple `cout` statements in a program
each is executed in order, top-to-bottom & left-to-right
- can have a single `cout` statement that spans multiple lines
more readable for long sentences (but can't break text in the middle)
- can have multiple `endl`'s to generate blank lines of output

program components: misc.

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

- C++ programs must be stored with a `.cpp` extension
- statements must end with a semi-colon
exceptions (since technically not statements): comments, `#include`, `{` and `}`
- the return statement must come last inside main
when the return statement is reached & executed, the program ends

program components: more misc.

```
// hello1.cpp      Dave Reed      8/24/01
//
// This simple program demonstrates C++ program structure.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221." << endl;

    return 0;
}
```

```
#include <iostream> using namespace std; int main(){cout<<
"Hello and welcome to CSC221."<<endl;return 0;}
```

- blank lines and indentation are irrelevant to the compiler
but help people to read the code & understand natural groupings of statements

which of the above would you rather read and debug?

syntax errors

```
// hello3.cpp      Dave Reed      8/24/01
//
// This bug-filled program demonstrates syntax errors.
////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    cout << "Hello and welcome to CSC221. << endl;

    return 0;
}
```

any violation of the format rules is called a *syntax error*

- will be caught by the C++ compiler when you try to compile the program
compiler will give you a descriptive error message, can click to identify
a single problem can cause multiple error messages → fix first error then try again
get used to errors, you will see them over & over (but try to learn from them!)

how many syntax errors in the above program?

input & variables

to read and process user input, first need a place to store the input
a *variable* is a memory location where a value is stored

- must declare a variable by specifying a type and a name

```
variableType variableName;
```

```
int age;                // creates variable named age to store an integer
double salary;         // creates variable named salary to store a real value
char initial;          // creates variable named initial to store a character
bool found;            // creates variable named found to store true/false
```

- wide range of numeric data types are provided, different ranges of values
integer types: short, **int**, long, ... real types: float, **double**, long double, ...
- variable names (a.k.a. identifiers) are sequences of letters, underscores,
and digits that start with a letter (*can also start with underscore, but don't*)
variable names are case-sensitive, so **age**, **Age**, and **AGE** are all different

```
double salary, bonus;    // can combine declarations using comma
```

input statements

`cin` statement reads in a value from the user & stores it in a variable

- `cin` is the standard C++ input stream (complements `cout`)
stream analogy: input value floats on stream, extract as it goes by
- the stream extraction operator `>>` is used to read values from the stream
can read multiple values from the stream using multiple instances of `>>`

```
cin >> variableName;
```

```
cin >> age; // reads integer value into age variable
cin >> salary >> bonus; // reads real values (salary first, then bonus)
```

- as a result of the `cin` statement, the value entered by the user is stored in the variable
any future reference to that variable will access that value

```
cout << "You entered " << age << " as your age." << endl;
```

input example

```
// age.cpp      Dave Reed      8/24/01
//
// This program reads the user's age and echoes it.
///////////////////////////////////////////////////////////////////

#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "Please enter your age: ";
    cin >> age;

    cout << "You entered " << age << " as your age." << endl;

    return 0;
}
```

before reading a value from the user, should display a prompt message

C++ strings

primitive data types for storing integers, reals, characters, & Booleans are built-in to C++

a *string* type has been added for storing text

- must include a library file that contains the string definition

```
#include <string>
```

- once this has been done, can declare and use variables of type string

```
string firstName;  
cout << "What is your name? ";  
cin >> firstName;  
  
cout << "Nice to meet you " << firstName << "." << endl;
```

- the string type has many useful operations associated with it
MORE LATER

string example

```
// greet.cpp      Dave Reed      8/24/01  
//  
// This program displays a personalized greeting.  
////////////////////////////////////  
  
#include <iostream>  
#include <string>  
using namespace std;  
  
int main()  
{  
    string firstName, lastName;  
    cout << "Enter your name (first then last): ";  
    cin >> firstName >> lastName;  
  
    cout << "Nice to meet you, " << firstName << " "<< lastName  
        << ". May I just call you " << firstName  
        << "?" << endl;  
  
    return 0;  
}
```

strings are delimited by whitespace
(i.e., arbitrary sequences of spaces,
tabs, and new-lines)

using Visual C++

I will assume you are programming using Visual C++

- available in G411 and numerous sites on campus

other C++ compilers exist (some free), but you are on your own

➤ to create a C++ program with Visual C++

1. Open Visual C++.
Start menu → Programs → Microsoft Visual Studio → Microsoft Visual C++
2. Create a project (collection of files that are compiled & linked together).
File → New → Win32 Console Application
must enter project name (e.g., hw1) and select location (e.g., a:)
will be prompted for default setup: select Empty Project
3. Create C++ source file and add to project.
File → New → C++ Source File
must enter file name (e.g., hw1.cpp) and verify project/location
4. Enter C++ program in file editor.
note: editor does color coding, automatic indentation, ...
save file by selecting: File → Save (or just Ctrl-S)

using Visual C++

➤ to execute a C++ program

1. Compile program
Build → Build hw1.exe (or just F7)
2. Status of compilation/linking will appear in window below.
if syntax errors occur, click on *first* error message
cursor will point to offending line in editor – fix & recompile.
if linker error occurs, check library names & verify "using namespace std;"
3. Execute program (but only after successful compilation/linking)
Build → Execute hw1.exe (or just Ctrl-F5)

➤ to open an existing project

1. Open Visual C++ (as before)
2. Open the project/workspace.
File → Open Workspace
browse to correct directory, select .dsw file

or just locate the directory for the project, and double click on the .dsw file