C++

Day 1 : Q&A

Q. What is the difference between a text editor and a word processor?

A. A text editor produces files with plain text in them. There are no formatting commands or other special symbols required by a particular word processor. Text files do not have automatic word wrap, bold print, italics, and so forth.

Q. If my compiler has a built-in editor, must I use it?

A. Almost all compilers will compile code produced by any text editor. The advantages of using the built-in text editor, however, might include the ability to quickly move back and forth between the edit and compile steps of the development cycle. Sophisticated compilers include a fully integrated development environment, allowing the programmer to access help files, edit, and compile the code in place, and to resolve compile and link errors without ever leaving the environment.

Q. Can I ignore warning messages from my compiler?

A. Many books hedge on this one, but I'll stake myself to this position: No! Get into the habit, from day one, of treating warning messages as errors. C++ uses the compiler to warn you when you are doing something you may not intend. Heed those warnings, and do what is required to make them go away.

Q. What is compile time?

A. Compile time is the time when you run your compiler, as opposed to link time (when you run the linker) or run-time (when running the program). This is just programmer shorthand to identify the three times when errors usually surface.

Quiz

1. What is the difference between an interpreter and a compiler?

2. How do you compile the source code with your compiler?

3. What does the linker do?

4. What are the steps in the normal development cycle?

Exercises

1. Look at the following program and try to guess what it does without running it.

```
1: #include <iostream.h>
2: int main()
3: {
4: int x = 5;
5: int y = 7;
6: cout "\n";
7: cout << x + y << " " << x * y;
8: cout "\n";
9:return 0;
10: }
```

2. Type in the program from Exercise 1, and then compile and link it. What does it do? Does it do what you guessed?

3. Type in the following program and compile it. What error do you receive?

```
1: include <iostream.h>
2: int main()
3: {
4: cout << "Hello World\n";
5: return 0;
6: }
```

4. Fix the error in the program in Exercise 3, and recompile, link, and run it. What does it do?

Day 2 : Q&A

Q. What does #include do?

A. This is a directive to the preprocessor, which runs when you call your compiler. This specific directive causes the file named after the word include to be read in, as if it were typed in at that location in your source code.

Q. What is the difference between // comments and /* style comments?

A. The double-slash comments (//) "expire" at the end of the line. Slash-star (/*) comments are in effect until a closing comment (*/). Remember, not even the end of the function terminates a slash-star comment; you must put in the closing comment mark, or you will get a compile-time error.

Q. What differentiates a good comment from a bad comment?

A. A good comment tells the reader why this particular code is doing whatever it is doing or explains what a section of code is about to do. A bad comment restates what a particular line of code is doing. Lines of code should be written so that they speak for themselves. Reading the line of code should tell you what it is doing without needing a comment.

Quiz

1. What is the difference between the compiler and the preprocessor?

2. Why is the function main() special?

3. What are the two types of comments, and how do they differ?

4. Can comments be nested?

5. Can comments be longer than one line?

Exercises

1. Write a program that writes I love C++ to the screen.

2. Write the smallest program that can be compiled, linked, and run.

3. BUG BUSTERS: Enter this program and compile it. Why does it fail? How can you fix it?

```
1: #include <iostream.h>
2: void main()
3: {
4: cout << Is there a bug here?";
5: }
```

4. Fix the bug in Exercise 3 and recompile, link, and run it.

Answers

Day 1

Quiz

1. What is the difference between interpreters and compilers?

Interpreters read through source code and translate a program, turning the programmer's code, or program instructions, directly into actions. Compilers translate source code into an executable program that can be run at a later time.

2. How do you compile the source code with your compiler?

Every compiler is different. Be sure to check the documentation that came with your compiler.

3. What does the linker do?

The linker's job is to tie together your compiled code with the libraries supplied by your compiler vendor and other sources. The linker lets you build your program in pieces and then link together the pieces into one big program.

4. What are the steps in the development cycle?

Edit source code, compile, link, test, repeat.

Exercises

1. Initializes two integer variables and then prints out their sum and their product.

2. See your compiler manual.

3. You must put a `#` symbol before the word `include` on the first line.

4. This program prints the words `Hello World` to the screen, followed by a new line (carriage return).

Day 2

Quiz

1. What is the difference between the compiler and the preprocessor?

Each time you run your compiler, the preprocessor runs first. It reads through your source code and includes the files you've asked for, and performs other housekeeping chores. The preprocessor is discussed in detail on Day 18, "Object-Oriented Analysis and Design."

2. Why is the function `main()` special?

`main()` is called automatically, each time your program is executed.

3. What are the two types of comments, and how do they differ?

C++-style comments are two slashes (`//`), and they comment out any text until the end of the line. C-style comments come in pairs (`/* */`), and everything between the matching pairs is commented out. You must be careful to ensure you have matched pairs.

4. Can comments be nested?

Yes, C++-style comments can be nested within C-style comments. You can, in fact, nest C-style comments within C++-style comments, as long as you remember that the C++-style comments end at the end of the line.

5. Can comments be longer than one line?

C-style comments can. If you want to extend C++-style comments to a second line, you must put another set of double slashes (`//`).

Exercises

1. Write a program that writes `I love C++` to the screen.

```
1: #include <iostream.h>
2:
3: int main()
4: {
5:    cout << "I love C++\n";
6:    return 0;
7: }
```

2. Write the smallest program that can be compiled, linked, and run.

```
int main(){}
```

3. BUG BUSTERS: Enter this program and compile it. Why does it fail? How can you fix it?

```
1: #include <iostream.h>
2: main()
3: {
4:      cout << Is there a bug here?";
5: }
```

Line 4 is missing an opening quote for the string.

4. Fix the bug in Exercise 3 and recompile, link, and run it.

```
1: #include <iostream.h>
2: main()
3: {
4:      cout << "Is there a bug here?";
5: }
```