Chapter 1:What is C++?

Profanity is the one language that all programmers understand

-- Anon

Review Questions

1. Define “class.”
2. Where are the computers that are to be used for homework for this course?
3. What compiler are we using for this course?

Chapter 2: The Basics of Program Writing

The first and most important thing of all, at least for writers today, is to strip language clean, to lay it bare down to the bone

Ernest Hemingway

Review Questions

1. Define “Machine Language.”
2. Define “Assembly Language.”
3. Define “source code.”
4. Define “object code.”
5. Define “library.”
6. Define “linker.”
7. Define “executable program.”
8. How is assembly language different from machine language?
9. What does a compiler do?
10. How is a compiler different from an assembler?
11. What is the extension for source files on our machine?
12. What type of program is used to create “source code.”
13. What is the extension for object files on our machine?
14. What type of programs are used to produce object files?
15. What is the extension for the executable programs?
16. What type of files go into creating an executable program?
17. What type of program is used to create an executable program?
18. How do you access the on-line help system for your compiler?

Chapter **3:Style**

There is no programming language, no matter how structured, that will prevent programmers from writing bad programs

- L Flon

It is the nobility of their style which will make our writers of 1840 unreadable forty years from now

- Stendhal

Review Questions

1. Why are comments *required* in a program?
2. Why must you write comments before or while you write the program, never after?
3. Which is better: 1) The underscore method of writing names, such as: `this_is_a_var`, or 2) The upper/lower case method: `ThisIsATest`?

Chapter 4: Basic Declarations and Expressions

A journey of a thousand miles must begin with a single step

Lao-zi

If carpenters made buildings the way programmers make programs, the first woodpecker to come along would destroy all of civilization

Anon

Review Questions

1. Name and define the three elements of a program.
2. Why are comments required in a program?
3. What is the purpose of the `return(0);` near the end of each program?
4. What punctuation character signals the end of each statement?
5. What statement to you use to print something to the screen?
6. What are the five simple C++ operators?
7. Evaluate the following C++ expressions:

   a) 5 + 3 / 2 + 1
   b) (5 + 3) / (2 + 1)
   c) 8 % (3 * 4) + 8 / 3 - 9
   d) 8 / (3 * 4 + 8 / 3 - 9)
   e) 4 + 9 * 6 - 8
   f) (11 % 7) / 3

8. What will the following statements print? (Assume i=5 and j=7.)

   a) `std::cout << i << '\n';`
   b) `std::cout << "i" << '\n';`
   c) `std::cout << i / j << '\n';`
   d) `std::cout << "i=" << i;`
   e) `std::cout << "i=" << i << '\n';`

9. Turn the equation $d=\frac{1}{2}\cdot g t^2$ into a C++ statement
10. What does it mean when you see the message “Warning: Null effect”?
11. Define “variable.”
12. Define “variable declaration.”
13. What are the three purposes for a variable declaration?

14. The FORTRAN language (among others) would automatically declare a variable the first time it was used Variables that began with A-H,O-Z where automatically declared **float** and variables that begin with I-M were automatically declared **int** Discuss the advantages and disadvantages of this feature (Note: FORTRAN was invented before lowercase was widely used in computes, so all variables were written in uppercase only)
15. Define `std::cout` and use it in a C++ statement.
16. Define "assignment statement" and give an example.
17. What is the difference between a real and an integer variable?
18. Give an example of an integer division statement and a floating point division statement?
19. The same operator '/' is used for both floating point and integer divides. How does the compiler know which one to use?
20. A thermos keeps hot things hot and cold things cold. How does it know which one to do?
21. The standard character set, ASCII, handles 128 characters. Only 95 of these can be typed in using the keyboard. What does a C++ programmer use to specify the none printing characters?
22. Define "character variable" and show how a character variable can be declared and used.
23. What are the possible value of a boolean (**bool**) variable?

Advanced Questions

The answers to these may require resources outside the scope of this book

24. What is the value of `4.5 % 3.2`?
25. Why are real numbers called floating point numbers?
26. What "floats" in a floating point number?
27. What is the biggest integer variable that you can have on your machine?
28. What is the biggest floating point variable that you can have on your machine?
29. Why are integer variables exact and floating point variables inexact?

Chapter 5: Arrays, Qualifiers, and Reading Numbers

That mysterious independent variable of political calculations, Public Opinion

Thomas Henry Huxley

Review Questions

1. What is the difference between an array and a simple variable?
2. What is the number of the last element of the following array?

```
int test[5];
```

3. What's the header file that's used to bring in the data definitions for C++ strings?

```
#include <string>
```

4. How do you concatenate two C++ style string?
5. What is the difference between a C style string and an array of characters?
6. Why must we use `std::strcpy` to assign one C style string to another?
7. We define a character array whose job is to hold a string What is the difference between the size of the array and the length of a C style string?
8. Can the size of any array change during the execution of a program?
9. Can the length of a string variable change during the execution of a program?
10. What happens if you try to read a C++ style string using `std::cin` and the `>>` operator? (Try it!) What about the C style strings?
11. How many elements are allocated for the multiple dimension array:

```
int four_dimensions[25][86][48][96];
```

Bonus question: Why will you probably never see a declaration like this in a real program?

12. Define the following:

long int	short int	unsigned	signed
float	double	register	auto
volatile	const	reference	extern

13. Why must you use the notation `static_cast<int>(very_short)` to write a very short number (a k a a character variable)? What would happen if you wrote it out without the `int` wrapper?
14. Define side effect.
15. Why are side effects bad things?

Chapter 6: Decision and Control Statements

Once a decision was made, I did not worry about it afterward

Harry Truman

Review Questions

1. Define the following:

a) Branching Statements
b) Looping Statements
c) if statement
d) else clause
e) relational operator
f) logical operator
g) while statement
h) Fibonacci number
i) break statement

Chapter 7: *The Programming Process*

Its just a simple matter of programming

Any boss who has never written a program

Review Questions

1. Describe the various steps of the programming process:
   - a Requirements
   - b Specification
   - c Code Design
   - d Coding
   - e Testing
   - f Debugging
   - g Release
   - h Maintenance
   - i Revision and updating
2. Why do we put each program in its own directory?
3. What is “Fast Prototyping?”
4. What is the *make* utility and how is it used?
5. Why is a “Test Plan” necessary?
6. What is “Electronic Archeology?”
7. What tools are available to aid you in going through some one elses code?
8. What is *grep* and how would you use it to find all references to the variable `total_count`?

Chapter 8: More Control Statements

Grammar, which knows how to control even kings

Moli re

Review Questions

1. Why do C++ programs count to five by saying “0, 1, 2, 3, 4?” Why should you learn to count that way?
2. Write a **for** statement to zero the following array:

   ```
   int data[10];
   ```

3. Why do we end each **case** with **break** or `// Fall through`?
4. Why do we always put a **break** at the end of each **switch**?
5. What will **continue** do inside of a **while**?
6. What will **continue** do inside of a **for**?
7. What will **continue** do inside of a **switch**? (This is a trick question)
8. What will **break** do inside of a **while**?
9. What will **break** do inside of a **for**?
10. What will **break** do inside of a **switch**? (This is *not* a trick question.)
11. Discuss the pros and cons of putting the **default** statement at the end of every **switch** Is there a case where putting the **default** statement somewhere else might be useful?

Chapter 9: Variable Scope and Functions

But in the gross and scope of my opinion
This bodes some strange eruption to our state

Shakespeare *Hamlet*, Act I, Scene I

Review Questions

1. Define “Variable Scope.”
2. Define “Variable Storage Class.”
3. Why are hidden variables a bad idea?
4. What is the scope of a function’s parameters?
5. What is the class of a function’s parameters?
6. The keyword **static** changes a local variable’s storage class from __________ to __________.
7. Why are all global variables permanent?
8. When is the following variable created, initialized, and destroyed?

```
int funct(void) {
    int var = 0     // The variable in question
    // ......
```

9. When is the following variable created, initialized, and destroyed?

```
int funct(void) {
    static int var = 0     // The variable in question
    // ......
```

10. When is the following variable created, initialized, and destroyed?

```
int var = 0     // The variable in question
int funct(void) {
    // ......
```

11. Define **namespace**.
12. What is the name of the namespace we've already used in this book for **cin** and **cout**?
13. Define “reference.”
14. What is binding and when does it occur?
15. What is the difference, if any, between the type of values returned by the following two functions:

```
int func1(void)
const int funct2(void)
```

16. What is the difference, if any, between the type of values returned by the following two functions:

```
int &fun1(void)
const int &fun2(void)
```

17. Which parameters in the following function can be modified inside the function:

```
void fun(int one, const int two, int &three, const int &four);
```

18. In the previous question, which parameters, when changed, will result in changes made to the caller's variables?

19. Which parameters in the following function can be modified inside the function:

```
void fun(int one[], const int two[]);
```

20. In the previous question, which parameters, when changed, will result in changes made to the caller's variables?

21. Define "Dangling Reference."

22. Why are dangling references a bad thing?

23. Can we overload the square function using the following two function definitions?

```
int square(int value);
float square(float value);
```

24. Can we overload the square function using the following two function definitions?

```
int square(int value);
float square(int value);
```

25. Define "default parameters."

26. Why must default parameters occur at the end of a parameter list?

27. What does the keyword **inline** do?

28. What programming technique was used to solve the "calculator problem" of Chapter 7, *The Programming Process*?

29. Define "Structured Programming."

30. Describe "Top down programming."

31. Describe "Bottom up programming."

32. Define "recursion."

33. What are the two rules you must follow to make a recursive function?

1) There must be an ending point. 2) Each stage of the function must make the problem simpler.

Chapter 10: The C++ Preprocessor

The speech of man is like embroidered tapestries, since like them this has to be extended in order to display its patterns, but when it is rolled up it conceals and distorts them

Themistocles

Review Questions

1. Define `#define`.
2. What is the difference between a simple #define macro and a **const** declaration?
3. When would you use conditional compilation?
4. Would you ever use conditional compilation for something other than debugging?
5. What is the difference between a parameterized macro and a simple macro?
6. What is the difference between a macro and a normal function?
7. What is the difference between a macro and an **inline** function?
8. Can the C++ preprocessor be run on things other than C++ code? If so what?
9. Why do we not want to use macros to define things like:

```
#define BEGIN {
#define END }
```

Chapter 11: Bit Operations

To be or not to be, that is the question

Shakespeare on boolean algebra

Review Questions

1. Describe the difference between *bitwise and* (&) and *logical and* (&&). Give an example of each.
2. Describe the difference between *bitwise or* (|) and *logical or* (||).
3. If you didn't have an *exclusive or* operator (^) how would you create one out of the other operators?
4. How would you test to see if any one of a set of bits is on? (This can be done without using the *logical and* (&&) operation.)
5. How would we rewrite the `set_bit` function if we used an array of **short int** instead of an array of **char**.

Chapter 12: Advanced Types

Total grandeur of a total edifice,
Chosen by an inquisitor of structures

Wallace Stevens

Review Questions

1. Define Structure.
2. Define Union.
3. How can we tell what type of data is currently stored in a union?
4. Define **typedef**.
5. Which is better **typedef** or **#define**? Why?
6. Define **enum** type.
7. List two reasons that **enum** types are better than defining a series of constants.
8. Define “bit field.”
9. What are the advantages and disadvantages of using bit fields.

Chapter 13: *Simple Classes*

She thinks that even up in heaven
Her class lies late and snores,

Cyril Connolly

Review Questions

1. Define “class”
2. Define the “big four” member functions.
3. What are the hidden functions called in the following code:

```
void use_stack(stack &local_stack)
{
    local_stack.push(9);
    local_stack.push(10);
}
```

4. Why can you overload the constructor and not overload the destructor?
5. Which of the big four are generated automatically?
6. How do you prevent each of them from being generated automatically if you don’t explicitly define one.?

Chapter 14: More on Classes

This method is, to define as the number of a class the class of all classes similar to the given class

Bertand Russell
Principles of Mathematics part II, chapter 11, section iii, 1903

Review Questions

1. Define “friend.”
2. Describe an example when you would use a “friend” function.
3. Define “constant member functions.”
4. Define “constant member variable.”
5. How do you initialize the constant member variables of a class?
6. Define “static” member variables.
7. Why are you able to access static member variables using the *class::var* syntax you cannot use the same syntax for ordinary member variables?
8. Define “static member function.”
9. Why are static member functions prohibited from accessing ordinary member variables?
10. Why can they access static member variables?

Chapter 15: *Simple Pointers*

The choice of a point of view is the initial act of culture.

Ortega y Gasset

Review Questions

1. Define "pointer."
2. How is a pointer different from a thing?
3. How many pointers can point to a single thing?
4. What happens if you assign a pointer NULL and then try and dereference it?
5. Are pointers more efficient than arrays?
6. Are pointers more risky to use than arrays?
7. Which of the preceding two questions is the more important?

Chapter 16: File Input/Output

I the heir of all the ages, in the foremost files of time

Tennyson

Review Questions

1. There are three different I/O packages described in this chapter: C++ streams, raw I/O, and C standard I/O Describe the advantages and disadvantages of each
2. Define `std::cin`, `std::cout`, and `std::cerr`.
3. Define `std::ifstream`.
4. Define `std::ofstream`.
5. How can we tell if a write worked?
6. How can we tell if a file exists?
7. What's the difference between `std::getline` and the `>>` *`string`* operation?
8. What does `sizeof` do?
9. Write a C++ program fragment to write out a variable containing the value "3" as "0003".
10. Why are graphic images almost always stored as binary files?
11. One type of image that's stored as an ASCII file is an X icon. Why is it ASCII?
12. What's the difference between 0 and '0'?
13. What type of end of line is used on your machine?
14. Someone took a text file from DOS and moved it to UNIX. When they tried to edit it a lot of control-M's (^M) appeared at the end of each line. Why?
15. Why is buffering a good idea?
16. When is it not a good idea?
17. Why is the C style `std::printf` more risky to use than the C++ style `std::cout`?

Chapter 17: Debugging and Optimization

Bloody instructions which, being learned, return to plague the inventor

Shakespeare on debugging

Review Questions

1. Describe useful debugging aides that might be built into a program
2. Describe your hardest bug and what you did to fix it.
3. What happens when you try to divide by 0?
4. What happens when you attempt to dereference the NULL pointer?
5. What happens when you modify the data pointed to by the NULL pointer?
6. What happens when you write a program that does infinite recursion?
7. Define `std::flush`..
8. Define “Confessional Method of Debugging.”
9. Define “Loop ordering.”
10. Define “Reduction in Strength.”
11. Write a program that divides an integer variable by 10 a thousand times and time how long it takes. How long does it take to divide the variable by 16?

Chapter 18: Operator Overloading

Overloaded, undermanned, ment to flounder, we
Euchred God Almightys storm, bluffed the Eternal Sea!

Kipling

Review Questions

1. Define “overloading.”
2. Define “operator overloading.”
3. What’s the difference between x++ and ++x?
4. How do you tell them apart when defining the operator functions?
5. What is casting?
6. Why should you not overload the casting operators? What should you use instead? Why?

Chapter 19: Floating Point

1 is equal to 2 for sufficiently large values of 1

Anonymous

Review Questions

1. Define “Overflow.”
2. Can you have overflow with integer calculations?
3. Define “Underflow.”
4. Can you have underflow in integer calculations?
5. What’s the biggest floating point number that can be represented on your machine.
6. On your pocket calculator what does 1-(1/3)-(1/3)-(1/3) equal?
7. Why do you never want to use floating point for money?

Chapter 20: Advanced Pointers

A race that binds
Its body in chains and calls them Liberty,
And calls each fresh link progress

Robert Buchanan

Review Questions

1. Define “Linked List.”
2. What would you use a linked list for?
3. Define “Double-linked List.”
4. What can a double-linked list be used for that a linked list cannot.
5. Define “binary tree.”
6. What are trees good for?
7. How would you delete an element from a tree?

Chapter 21: Advanced Classes

The ruling ideas of each age have ever been the ideas of its ruling class

Karl Marx

Manifesto of the Communist Party

Review Questions

1. Define “derived class.”
2. Define “base class.”
3. Can a class be both derived and base?
4. Define “virtual function.”
5. Define “pure virtual function.”
6. Describe how to implement virtual functions using function pointers. (This is what C++ does internally.)
7. Define “abstract class.”
8. Why can’t you use an abstract class to define a variable?

Chapter 22: Exceptions

How glorious it is and also how painful to be an exception

Alfred De Musset

Review Questions

1. Define “exception.”
2. Define “try.”
3. Define “catch.”
4. Define “throw.”

Chapter 23: Modular Programming

Many hands make light work

John Heywood

Review Questions

1. Define “header file.”
2. What should go in a header file? Why?
3. What should not go in a header file? Why?
4. Define “extern.”
5. Define “static.” (Warning: It’s used for a lot of things.)
6. Define “infinite array.”

Chapter 24: Templates

Thou cunningst pattern of excelling nature,

— Shakesphere, *Othello Act V*

Review Questions

1. Define “template.”
2. When does a template cause code to be generated?
3. What is a function specialization?
4. What is a class specialization?
5. Define "export"?
6. Name a compiler which implements the "export" directive correctly.

Chapter 25: Standard Template Library

Goodness and evil never share the same road, just as ice and charcoal never share the same container.

— Chinese proverb

Review Questions

1. Define “STL Container.”
2. What are the differences between a STL vector and a C++ array?
3. How do you check to see if an iterator has reached the end of a container?
4. What's stored in a map?

Chapter 26: Program Design

If carpenters made houses the way programmers design programs, the first woodpecker to come along would destroy all of civilization

— Traditional computer proverb

Review Questions

1. Name and define three design goals that should be a part of any program's design.
2. What are other design factors that might be considered.
3. MS/DOS was designed for a 640K PC with 360K floppies as it's primary storage device. Did the design of the operating system make it easy to expand the system for today's modern computers?
4. What makes a good module?
5. What is information hiding?
6. Why are global variables considered bad?
7. Are all global variables bad?
8. One of the problems with C++ is that a class must expose implementation information to modules using that class.   Where is this implementation information located?

Chapter 27: Putting It All Together

For there isnt a job on the top of the earth the beggar dont know, nor do
Kipling

Review Questions

1. Describe the programming process used to create the program created in this chapter. Explain what worked and what didn't. If something didn't work, what can be done to fix the problem?

Chapter 28: From C to C++

No distinction so little excites envy as that which is derived from ancestors by a long descent

François De Saliganc De La Mothe Fénelon

Review Questions

1. Define “K&R Style Functions.”
2. Define `malloc`.
3. Define `free`.
4. Why should you never use `malloc` on a class?
5. Why should you never use `free` on a class?
6. How do you initialize a structure to be all zeros? How do you initialize a class to be all zeros?
7. Define `setjmp` and `longjmp`.
8. What are the dangers associated with using these functions?

Chapter 29: Programming Adages

Second thoughts are ever wiser

Euripides

No review questions

Supplement: From C to C++

New nobility is but the act of power, but ancient nobility is the act of time

Francis Bacon

Review Questions

1. Why are comments *required* in a program?
2. Why must you write comments before or while you write the program, never after?
3. Which is better: 1) The underscore method of writing names example: `this_is_a_var`, or the upper/lower case method: `ThisIsATest`? (Note this is a religious issue and has no right answer.)
4. Define `std::cout` and use it in a C++ statement.
5. Define the following:

long int	short int	unsigned	signed
float	double	register	auto
volatile	const	reference	extern

6. Why must you use the notation `int(very_short)` to write a very short number (aka. a character variable)? What would happen if you wrote it out without the `int` wrapper?
7. Define side effect.
8. Why are side effects bad things.
9. When is the following variable created, initialized and destroyed?

```
int funct(void) {
    int var = 0     // The variable in question
    // ......
```

10. When is the following variable created, initialized and destroyed?

```
int funct(void) {
    static int var = 0     // The variable in question
    // ......
```

11. When is the following variable created, initialized and destroyed?

```
int var = 0     // The variable in question
int funct(void) {
    // ......
```

12. Define "reference"?
13. What is binding and when does it occur?

14. What is the difference, if any, between the type of values returned by the following two functions:

```
int func1(void)
const int funct2(void)
```

15. What is the difference, if any, between the type of values returned by the following two functions:

```
int &fun1(void)
const int &fun2(void)
```

16. Which parameters in the following function can be modified inside the function:

```
void fun(int one, const int two, int &three, const int &four);
```

17. In the previous question, which parameters when changed will result in changes made to the caller's variables?

18. Which parameters in the following function can be modified inside the function:

```
void fun(int one[], const int two[]);
```

19. In the previous question, which parameters when changed will result in changes made to the caller's variables?

20. Define "Dangling Reference."

21. Why are dangling references a bad thing?

22. Can we overload the square function using the following two function definitions?

```
int square(int value);
float square(float value);
```

23. Can we overload the square function using the following two function definitions?

```
int square(int value);
float square(int value);
```

24. Define "default parameters."

25. What does the keyword **inline** do?

26. What programming technique was used to solve the "calculator problem" of Chapter 7, *The Programming Process.*