Chapter 13

Object Oriented Programming Using C++

1. C++ was originally developed by
(a) Clocksin and Mellish
(b) Donald E. Knuth
(c) Sir Richard Hadlee
(d) Bjarne Stroustrup

***2.** `cfront`
(a) is the front end of a C compiler
(b) is the pre-processor of a C compiler
(c) is a tool that translates a C++ code to its equivalent C code
(d) none of the above

***3.** The following program fragment

```
int i=10;
void main( )
{
    int i=20;
    {
       int i=30.;
       cout << i << ::i;
    }
}
```

(a) prints `3010`
(b) prints `3020`
(c) will result in a run time error
(d) none of the above

***4.** Which of the following are procedural languages?

(a) Pascal (b) Smalltalk (c) C++ (d) C

***5.** A function `abc` is defined as

```
void abc(int x=0, int y=0)
{ cout << x << y; }
```

Which of the following function calls is/are illegal? (Assume `h,g` are declared as integers)

(a) `abc();` (b) `abc(h);` (c) `abc(h,h);` (d) None of the above

6. The following C++ code results in

```
#include "iostream.h"
void main(void)
{
        cout << (int i=5) << (int j=6);
}
```

(a) compilation error
(b) run time error
(c) link time error
(d) none of the above

***7.** Reusability is a desirable feature of a language as it

(a) decreases the testing time (b) lowers the maintenance cost
(c) reduces the compilation time (d) reduces the execution time

***8.** Choose the correct statements regarding `inline` functions.

(a) It speeds up execution (b) It slows down execution
(c) It increases the code size (d) It decreases the code size

9. If many functions have the same name, which of the following information, if present, will be used by the compiler to invoke the correct function to be used?

(a) The operator : : (b) The return value of the function
(c) Function signature (d) None of the above

***10.** The statements

```
int a = 5;
cout << "FIRST" << (a<<2) << "SECOND";
```

outputs

(a) `FIRST52SECOND` (b) `FIRST20SECOND`
(c) `SECOND25FIRST` (d) an error message

11. Choose the correct remarks.

(a) C++ allows any operator to be overloaded.
(b) Some of the existing operators cannot be overloaded.
(c) Operator precedence cannot be changed.
(d) All of the above.

12. A constructor is called whenever

(a) an object is declared
(b) an object is used
(c) a class is declared
(d) a class is used

***13.** Which of the following remarks about the differences between constructors and destructors are correct?

(a) Constructors can take arguments but destructors cannot.
(b) Constructors can be overloaded but destructors cannot be overloaded.
(c) Destructors can take arguments but constructors cannot.
(d) Destructors can be overloaded but constructors cannot be overloaded.

***14.** The following program fragment

```
void main( )
{
    int x=10;
    int &p=x;
    cout << &p << &x;
}
```

(a) prints 10 and the address of x
(b) results in a run time error
(c) prints the address of x twice
(d) prints the address of p twice

***15.** The declaration

```
int x; int &p=x;
```

is same as the declaration

```
int x,*p; p=&x;
```

This remark is

(a) true (b) false (c) sometimes true (d) none of the above

16. The following program segment

```
const int m=10;
int &n=m;
n=11;
cout << m << n;
```

(a) results in compile time error
(b) results in run time error
(c) prints 1111
(d) prints 1011

***17.** The following program segment

```
int a=10;
int const &b=a;
a=11;
cout << a << b;
```

(a) results in compile time error
(b) results in run time error
(c) prints 1111
(d) none of the above

The next three questions are based on the following information.

```
int a=1, b=2;
a = chg(b);
cout << a << b;
```

***31.** If the function `chg` is coded as

```
int chg(int x)
{
    x = 10;
    return (11);
}
```

then

(a) it results in compile-time error
(b) it results in run time error
(c) it prints `112`
(d) it prints `1110`

***32.** If the function `chg` is coded as

```
int chg(int &x)
{
    x = 10;
    return(11);
}
```

then

(a) it results in compile-time error
(b) it results in run time error
(c) it prints `112`
(d) it prints `1110`

***33.** If the function `chg` is coded as

```
int chg(const int &x)
{
    x=10;
    return(11);
}
```

then

(a) it results in compile-time error
(b) it results in run time error
(c) it prints `112`
(d) it prints `1110`

34. Choose the correct statements from the following:

(a) In a struct, the access control is public by `default`.
(b) In a struct, the access control is private by `default`.
(c) In a class, the access control is public by `default`.
(d) In a class, the access control is private by `default`.

the output will be

(a) `There was There was`

(b) nothing

(c) `There was There was a certain man`

(d) `There was a certain man There was a certain man`

The next two questions are based on the following program segment.

```
class A
{ int i1;
   protected:
    int i2;
   public:
    int i3;
} ;
class B : public A
{ public:
  int i4;
};
class C : B
{};
```

48. The variable `i2` is accessible

(a) to a public function in `class A`

(b) to a public function in `class B`

(c) to a public function in `class C`

(d) from the main function

49. Which variable(s) is/are accessible from the main function?

(a) `i1`

(b) `i2`

(c) `i3`

(d) None of the above

***50.** The following program

```
class abc;
class def
   {   int i1;              // statement 1
       protected: int i2;   // statement 2
       public: int i3;      // statement 3
   friend abc;
   };
   class abc
   { public:
       void main(def A)
       {cout << (A.i1=3); cout << (A.i2=4); cout << (A.i3=5)}
   };
```

```
void main( )
{
        def x1;
        abc x2;
        x2.mn(x1);
}
```

(a) will compile successfully if statement 1 is removed
(b) will compile successfully if statement 2 is removed
(c) will compile successfully if statement 3 is removed
(d) will run successfully and print 345

The next two questions are based on the following C++ program.

```
# include "iostream.h"
int a(int m)
{ return ++m; }
int b(int &m)
{ return ++m; }
int c(char &m)
{ return ++m; }
void main( )
{
  int p=0, q=0, r=0;
  p += a(b(p));
  q += b(a(q));
  r += a(c(r));
  cout << p << q << r;
}
```

51. The above program
(a) results in compilation error
(b) prints 123
(c) prints 111
(d) prints 322

52. If the statement
q += b(a(q)); is replaced by the statement
q += b(a(p)); then the above program
(a) prints 111
(b) results in compilation error
(c) prints 322
(d) prints 352

53. Consider the declarations
char a;
const char aa = 'h';
char *na;

const char *naa;

Which of the following statements

Statement I: aa = a;

Statement II: na = &a;

Statement III: na = &aa;

is/are illegal?

(a) Only I and II (b) Only II and III

(c) Only I and III (d) All the three statements are illegal

54. Forgetting to include a file (like cmath or math.h) that is necessary will result in

(a) compilation error (b) warning when the program is run

(c) error at link time (d) warning when the program is compiled

***55.** Assume that the random number generating function – rand( ), returns an integer between 0 and 10000 (both inclusive). If you want to simulate the throwing of a die using this random function, use the expression

(a) rand( ) % 6 (b) rand( ) % 6 + 1

(c) rand( ) % 5 + 1 (d) none of the above

***56.** Assume that the random number generating function – rand( ), returns an integer between 0 and 10000 (both inclusive). To randomly generate a number between a and b (both inclusive), use the expression

(a) rand( ) % (b-a) (b) (rand( ) % a) + b

(c) (rand( ) % (b-a)) + a (d) (rand( ) % (b-a+1)) + a

57. Which of the following comments about inline comments are true?

(a) A function is declared inline by typing the keyword *inline* before the return value of the function.

(b) A function is declared inline by typing the keyword *inline* after the return value of the function.

(c) A function that is declared inline may not be treated inline.

(d) Inline functions are essentially same as implementing a function as macro.

58. Which of the following decides if a function that is declared inline is indeed going to be treated inline in the executable code?

(a) Compiler (b) Linker (c) Loader (d) Preprocessor

***59.** Which of the following type of functions is an ideal candidate for being declared *inline*?

(a) A function that is small and is not called frequently.

(b) A function that is small and is called frequently.

(c) A function that is not small and is not called frequently.

(d) A function that is not small and is called frequently.

***60.** One of the disadvantages of pass-by reference is that the called function may inadvertently corrupt the caller's data. This can be avoided by

(a) passing pointers (b) declaring the formal parameters constant

(c) declaring the actual parameters constant (d) all of the above

46. d	47. c	48. a, b, c	49. d	50. d
51. d	52. d	53. c	54. c	55. b
56. d	57. a, c	58. a	59. b	60. b
61. d	62. c	63. b	64. a, b	65. a, b
66. d				

Explanations

2. When C++ was developed. it did not have a compiler. It used `cfront`, which translated the C++ code into a C code and then the existing C compiler was used to compile the program originally written in C++.

3. `::` is basically meant to manipulate a global variable, in case a local variable also has the same name.

4. Procedural languages sequentially execute a set of imperative statements to achieve the desired effect. Most of the traditional languages fall in this category.

5. Both the arguments are optional. All the calls are legal.

7. Reusable code is an already used code, as the name implies. Hence it is bug-free and pre-tested. There is no need to test it.

8. Each occurrence of the `inline` function call will be replaced by its body. No function call overhead will be there but the size of the code will increase.

10. The symbol << has a context sensitive meaning. The << in (`a<<2`) means shifting `a` by 2 bits to the left, which is nothing but multiplying it by 4. So, `a<<2` will be $5 \times 4 = 20$ and hence the output will be `FIRST20SECOND`.

13. Since destructors do not take arguments, the question of overloading does not arise at all.

14. `int &p=x` aliases `p` to `x`. This means they refer to the same memory location. So, the address of `x` will be same as that of `p`.

15. In the first declaration, `p` is aliased to `x`. In the second case, `p` is a separate variable and it will have an address that is different from that of `x`.

17. The very idea of declaring `b` as a constant integer is to protect it from changes. However, since it is aliased to a variable whose attribute is not `const`, the value of `b` can be indirectly changed by changing the value of `a`. This is a bad programming practice.

19. C++ forbids initialization with strings, whose length is more than the size of the array. A *C* compiler permits.

26. Since the second argument is mandatory, any call should have at least the first two parameters. Some compilers expect the optional parameters to follow the others. Such compilers give a compilation error.

31. Here the `default` parameter passing mechanism of pass by value will be used. Any change done to the parameter will not be reflected outside the function. So, `b` retains its value `2`.

32. Here the parameter passing mechanism is pass by reference. Any change done to the parameter will be reflected outside the function. So, the value of `b`, after the `exit` of the function will be `10`.