1. What is the major difference in C++ comparing with C
    a) Functions
    b) Pointers
    c) OOP's concepts
    d) All the above

Ans: C

Explanation: C++ is an intermediate language; it supports low-level language and high-level language. So, it supports the OOP's concept and C is low level language.

2. C++ follows _____ approach.
    a) Top-down
    b) Bottom-up
    c) Both
    d) None of the above

Ans : Bottom approach

Explanation: C++ having the objects, these objects are formed the top-level system by linking with other classes. But in C language starts from main() function goes to the last statement of the main(). So, this approach is called top-down approach.

3. What is function overloading?

Ans: Function overloading allows creating several functions having same name which are different from each other in the type of input and output. (i.e. Functions having same name with different signatures.)

Syntax: return_type function_name(arguments)

```
{
    //statements;
    return value;
}
```

4. How variable declaration in C++ differs from C?

Ans: C requires all the variables to be declared at the beginning of a scope but in C++ we can declare variables anywhere in the scope. This makes the programmer easier to understand because the variables are declared in the context of their use.

5. What are tokens in C++?

Ans: The smallest individual units of a program are known as tokens. C++ has the following tokens: Keywords, Identifiers, Constants, Strings, and Operators.

6. The additional use of **void** is_____

Ans: To declare the generic pointer.

7. C++ places no limit on its length and all the characters in a name are significant.
    a) True

b) False

Ans: a-True.

8. **Constants** refer to fixed values that change during the execution of a program.
   a) True
   b) False

Ans: False

9. What is the character set that cannot fit a character into a single byte?
   a) wchar_t
   b) int
   c) float
   d) setw

Ans: A

Explanation: The wchar_t type is a wide-character literal and it is intended for characters sets that cannot fit a character into a single byte. Wild-character literals begin with the letter L.

10. What are the Built-in type from the following?
    a) int
    b) float
    c) char
    d) void

Ans: d

Explanation: The built in data types are also known as basic data types. Basic data types may have several modifiers preceding them to serve the needs of various situations.

11. What are the user-defined data types
    a) structures
    b) union
    c) Both a and b
    d) int

Ans: C

12. Does C++ supports the type compatibility?

Ans: yes

13. The code below declares and define variable x

extern int x;

a) True
b) False

Ans: false

Explanation: An extern declaration does not define the variable unless it is also initialized in the same statement.

14. What is the value of the local variable x at the end of main?

```
#include <iostream>
using namespace std;
int x = 5;
int main(int argc, char** argv)
{
   int x = x;
   return 0;
}
```

a) 5
b) 0
c) Undefined
d) None

Ans: c

Explanation: The local x in main hides the global x before the local x's initializer is considered. Therefore the local x is being initialized with itself (the local unitialized variable)

15. What is the value of y at the end of main?

```
const int x = 5;
int main(int argc, char** argv)
{
   int x[x];
   int y = sizeof(x) / sizeof(int);
   return 0;
}
```

a) 0
b) 5
c) 20
d) Undefined

Ans: b

Explanation: The local x does not hide the global x until the end of the declaration. A local name will hide a global after the end of the declaration but before the beginning of initialization.

16. What is the output of the following?

```
#include <iostream>
using namespace std;
int main()
```

```
{
    typedef int num;
    num a = 10, b = 15;
    num c = a + b + a - b;
    cout << c;
   return 0;
}
```

a) 20
b) 15
c) 30
d) 25

Ans: a

Explanation: In this program, we are manipulating the numbers and printing the result using user-defined data types.

17. Which of the following statements is NOT valid about operator overloading?
    a) Only existing operators can be overloaded.
    b) Overloaded operator must have at least one operand of its class type.
    c) The overloaded operators follow the syntax rules of the original operator.
    d) None of the mentioned

Ans: a

Explanation: In this program, -> operator is used to describe the member of the class.

18. Operator overloading is
    a) making C++ operator works with objects
    b) giving new meaning to existing operator
    c) making new operator
    d) both a & b

Ans: d

Explanation: Operator overloading is the way of adding operation to the existing operators.

19. What is the output of following program

```
#include <iostream>
using namespace std;
ostream & operator<<(ostream & i, int n)
{
        return i;
}
int main()
{
  cout << 5 << endl;
  cin.get();
```

```
    return 0;
}
```

a) 5

b) 6

c) Error

d) Runtime error

Ans: c

Explanation: In this program, there will arise an ambiguous overload for 5.

20. How to declare operator function?

a) Operator operator sign

b) Operator

c) operator sign

d) None of the mentioned

Ans: a

Explanation: We have to declare the operator function by using operator, operator sign. Example “operator +” where operator is a keyword and + is the symbol need to be overloaded.

21. What is the output of following program

```
#include <iostream>
using namespace std;
class sample
{
   public:
   int x, y;
   sample() {};
   sample(int, int);
   sample operator + (sample);
};
sample::sample (int a, int b)
{
   x = a;
   y = b;
}
sample sample::operator+ (sample param)
{
   sample temp;
   temp.x = x + param.x;
   temp.y = y + param.y;
   return (temp);
}
int main ()
{
   sample a (4,1);
   sample b (3,2);
   sample c;
   c = a + b;
   cout << c.x << "," << c.y;
```

```
        return 0;
  }
```
Ans: Output 7,3

22. Which of the following operators can't be overloaded?
    a) ::
    b) +
    c) –
    d) []

Ans: a

23. What is inline function?

Ans: An inline function is a function that expanded in line when it is invoked i.e. the compiler replaces the function call with the corresponding function code.

24. What are the situations where inline expansion may not work?
    a) For functions returning values, if a loop, a switch, or a goto exists.
    b) If inline functions are recursive.
    c) If functions contain static variables.
    d) All the above

Ans: d

Explanation: The inline keyword merely sends a request, not a command, to the compiler.

The compiler may ignore this request if the function definition is too long or too complicates and compile the function as a normal function.

25. What is function overloading?

Ans: The same function name to create functions that perform a variety of different tasks.

26. What are the rules for function overloading?
    a) Function name should be same.
    b) The function arguments differ from data types.
    c) Number of arguments should be different.
    d) All the above.

Ans: d

27. The Return type of function is_____ by default

Ans: int

28. When a function is declared inline, the complier replaces the function call with the respective function code, normally, a small size function is made as _____

Ans: inline.

29. What is the output of following program

```
#include<iostream.h>
int fun()
```

```
{
       return 1;
}
float fun()
{
      return 10.23;
}
void main()
{
        cout<<(int)fun()<<' ';
        cout<<(float)fun()<<' ';
}
```

Ans: Type mismatch.