Celect

PART
« B »

Data Types

- In little-endian systems addresses are always in ABCD and data is stored in DCBA order
- In big-endian system addresses are always in ABCD and data is stored in ABCD order
- Cycle is present in integer and character data type but not present in float or double types
- Increments the value of a float or double variable beyond its maximum range that is +INF and beyond it minimum range is –INF
- The minimum octal character constant is '\000' and maximum octal character constant is '\377'
- Very first escape sequence character is '\a' and last escape sequence character is '\r'
- Float data always stores in memory mantissa and exponent format
- Enum data types creates a sequence sets of integral constants
- There is no cycle present in enum data type
- BCPL is a type less language
- When a language is able to produce a new data type that is called extensibility
- Tyepdef creates a new name but does not create a new type
- Addresses in memory always in ABCD format but data store in memory in ABCD or DCBA that depends on system
- The process of byte ordering is known as endianess
- 32 bits recurring binary of a float is always lesser than 64 bits recurring binary of a float
- When a signed negative integer compared with an unsigned integer , its binary level of variable is compared not their value level
- Signed and unsigned modifier is not allowed in float or double data types
- All constants in C are Rvalue category of objects
- Constant variables, array name, function name, enum constants are Rvalue category of objects
- All escape sequence characters are octal character constants
- The size of the null string constant is 1 byte
- '\0' is null character constant whose ASCII value is 0
- Length of the Variable name beyond 32 characters are no use

1. Write the missing statement to display 19 on the screen

```
void main()
{
   char x='19';
   printf("%X",x);
}
```

2. Debug the program to get the output 300

```
void main()
{
   int x;
   x=300*300/300;
   printf("%d",x);
}
```

3. Find the output

```
void main()
{
   float a=4;
   int i=2;
   printf("%f %d",i/a,i/a);
   printf("%d %f",i/a,i/a);
}
```

4. What is the problem in the following code?

```
void main()
{
   float f=5,g=10;
   enum{i=10,j=20,k=50};
   printf("%d\n",++k);
   printf("%f\n",f<<2);
   printf("%lf\n",f%g);
   printf("%lf\n",fmod(f,g));
}
```

5. Find the output

```
void main()
{
   char ch=291;
   printf("%d%d%c",32770,ch,ch);
}
```

6. Find the output

```
void main()
{
   int i=10,j=11,k=12;
   printf("%X%x%ci%x",j,i,'s',k);
}
```

7. Find the output.

```
void main()
{
   printf("%d\n",'-'-'-'-'/'/'/');
}
```

8. Find the output

```
void main()
{
   int x=10,y=5,p,q;
   p=x>9;
   q=x>3&&y!=3;
   printf("p=%d q=%d",p,q);
}
```

9. What is the problem in the following code?

```
float x=9.0;
main()
{
   unsigned float x=8.0;
   if(x=x)
      printf("Hello");
   else
      printf("Hi");
}
```

10. Find the output

```
void main()
{
   unsigned volatile static
                  const int i=-5;
   +i;
   printf("%d",i);
   printf("%d",sizeof(i));
}
```

11. Find the output.

```
void main()
{
   double a=11.11,b;
   int *ptr;
   ptr=(int*)&a;
   b=*ptr;
   printf("%f",*ptr);
}
```

12. Find the output.

```
void main()
{
   int a=6;
   float b=3.000000f;
   b=b*2;
   if(a==b)
   {
      a=a+b;
      printf("%d",a);
   }
   else
   {
      a=a-b;
      printf("%d",a);
   }
}
```

13. Find the output

```
void main()
{
   if(~0==(unsigned int)-1)
      printf("%d %u",~0,(unsigned int)-1);
   else
      printf("%d",~1);
}
```

14. Find the output

```
void main()
{
   printf("%s","C Marathon"+2);
}
```

15. Find the output

```
int main()
{
   unsigned int a=-1;
   int b;
   b=~0;
   if(a==b)
      printf("equal");
   else
      printf("not equal");
      return 0;
}
```

16. Find the output

```
void main()
{
   printf("%d"+0,145);
   printf("%f"+1,145);
}
```

17. What is the problem in the following code?

```
void main()
{
   int const *p=5;
   int q;
   p=&q;
   printf("%d",++(*p));
}
```

18. What is the problem with the following code?

```
typedef enum error Type
{
   warning,error,exception,
}error;
void main()
{
   error g1;
   g1=1;
   printf("%d",g1);
}
```

19. Find the output

```
void main()
{
   int const i=5;
   int *p;
   p=&i;
   printf("%d",*(p++));
}
```

20. Find the output

```
void main()
{
   int const i=5;
   int const *p;
   p=&i;
   printf("%d",*(p++));
}
```

21. What is the problem in the following code?

```
void main()
{
   int i=5;
   int const *p;
   p=&i;
   printf("%d",(*p)++);
}
```

22. What is the problem in the following code?

```
void main()
{
   int i=5;;;
   int j=24;;;
   printf("%d %d",i,j);;;;
}
```

23. What is the problem in the following code?

```
void main()
{
   int i=5;
   int j=24;;
   printf("%d %d",i,j);;
}
```

24. What is the problem in the following code?

```
void main()
{
   void a=5;
   printf("%d",a);
}
```

25. Find the output

```
void main()
{
   printf("\nab");
   printf("\bsi");
   printf("\rha");
}
```

26. What is the problem in the following code?

```
enum outside
{
   sunday,
   monday,
   tuesday
};
void main()
{
   enum inside
   {
      wednesday,
      thursday,
      sunday
   };
   printf("%d",sunday);
}
```

27. Why condition is false in the following statement?

```
float x=4.6;
if(x==4.6)
   printf("Hello");
else
   printf("hi");
```

28. What is the problem in the following code?

```
void main()
{
   char x='\477';
   printf("%c",x);
}
```

29. What is the problem in the following code?

```
void main()
{
   enum xxx{a=32765,b,c,d};
}
```

30. Write the missing statement so that the output will be brainq25 6.

```
void main()
{
   int a,b;
   a=25,86,74;
   b=/*add here*/
   printf("%d %d",a,b);
}
```

31. Write the missing statement so that the output will be 0 1 2. There are four such statements write all the four.

```
void main()
{
   /*add here*/
   printf("%d %d %d",BLACK,BLUE,GREEN);
}
```

32. How many times the following loop will continue

```
void main()
{
   short int x=1;
   while(x)
   {
      printf("C Marathon");
      x++;
   }
}
```

33. Write one line statement to change the byte order of an integer,(means first byte will be second byte and second byte will be first byte)

34. Find the output

```
void main()
{
   char ch;
   for(ch='0';ch<='\377';ch++)
   printf("%d",ch);
}
```

35. How many times the following loop will continue?

```
void main()
{
   int i=1;
   for(i=1;i<=40000;i++)
      printf("\nHello bbsr");
}
```

36. Find the output

```
void main()
{
   int zero,zero1;
   zero=10/0;
   zero1=0/0;
   printf("%d %d",zero,zero1);
}
```

37. Write the source code to check the system is little-endian or big-endian system.

38. Design a program that will calculate the sizeof any variable.

39. Debug the following program

```
#include"ganga.h"
void main()
{
   show();
}
int const x=80;
//ganga.h
void show()
{
   extern int x++;
   printf("%d",x);
}
```

40. Give the memory representation of float x=20.50

41. Write the missing statement to debug the program.

```
void main()
{
   int a;
   a=f(10,3.14);
   printf("%d",a);
}
f(int aa,float bb)
{
   return((float)aa+bb);
}
```

42. How to check, cycle is present or not in integer data type.

43. Write efficient algorithm to find number of bits are set to 1 in 16 bits short integer (checking each bit using condition is not the solution)

44. Find the output

```
void main()
{
   const int x=get();
   printf("%d",x);
}
get()
{
   return(20);
}
```

45. This is a program to count number of bits set to 1. But this only works for positive number. Modify the program or redesign the program to do the same for negative numbers also.

```
void main()
{
   int x;
   int c=0;
   printf("\nEnter any number :");
   scanf("%d",&x);
   while(x>=1)
   {
      c++;
      n=n&n-1;
   }
   printf("%d",c);
}
```

46. In which line of the following program it will show error.

```
1. void main()
2. {
3.    void *p;
4.    int **p1;
5.    int a;
6.    a=129;
7.    p=&a;
8.    p1=&p;
9.    printf("*p=%d\n\n",*p);
10.   printf("p=%d\n\n",p);
11.   printf("&p=%d\n\n",&p);
12.}
```

47. Find the output

```
void main()
{
   int x=300;
   printf("%d",(char)x/2);
}
```

48. Find the output

```
void main()
{
   int x=30000;
   x=x+3000;
   printf("%d",x);
}
```

49. Find the output

```
void main()
{
   float x=4.3;
   if(x==4.3)
      printf("Hello");
   else
      printf("Hi");
}
```

50. Find the output.

```
void main()
{
   int a=0x1234;
   a=a>>12;
   a=a<<12;
   printf("%x",a);
}
```

ANSWERS

1. **Ans:** char x='\x19'

   Explanation : '%x' is a format specifier to print a hexadecimal char constant. In order to write a hexadecimal char const, we have to write '\x' preceding the char const. so in order to get the output 19, we have to write char x='\x19' at time of defining.

2. **Ans:** x=300l*300/300

   Explanation : In order to get the output 300, we have to write either x=300l*300/300 or x=300*300L/300. 300 is a int but 300*300 result in long which can not be stored in an integer which is of 2 bytes. so if we make any of the two 300 as long int by writing 300L, then the result will be a long int in which 90000 can be stored perfectly. When an operation is done between two different data types of different size, then the shorter data type gets converted to the higher data type and the result also belongs to the higher data type.

3. **Ans:** 0.500000 0 0 0.000000.

   Explanation : as 'a' is of float datatype and 'i' is of int datatype, so the result of i/a will definitely be a float. In the case of 1st printf(), the 1st format specifier is %f so the result gets printed in float and as the 2nd format specifier is %d, the result gets printed in integer format so the output of 1st printf() is 0.500000and 0.but in the 2nd case, the 1st format specifier is %d and hence theresult gets converted to an int whose value is 0, after that for the 2nd format specifier %f, the previous result will get converted to an float whose value is 0.000000. Hence the output of the 2nd printf() will be 0 and 0.000000.

4. **Ans:** Compilation error.

   Explanation : The individual members in an enum data type are constants and hence their value can not be changed (so ++k is illegal). The operators '<<' and'%' work with integral data type only. The prototype of fmod() is present inside "math.h" and the purpose of this function is to return the remainder where both the divisor and the dividend belong to float data type.

5. **Ans:** -32766 0 #

   Explanation : 32770 is a long int which will take 4 bytes to get stored in the memory, its memory representation will be 11111111 00000000. but in the printf() statement, the 1st format specifier is %d and hence it will take 1st 2 bytes of data and print -32766 and the 2nd %d will take the rest 2 bytes of data whose value is zero. As all the integral datatype maintain a cycle internally in the memory, the value stored in ch will be converted to 35 which is the ASCII value of '#'.

6. **Ans:** Basic

   Explanation : %x is the format specifier for the hexadecimal char constant, simply int i ,j and k will get converted to its equivalent hexadecimal constants which is nothing but B a and c respectively. So the final output will be Basic.

7. Ans: -1

Explanation : Anything written within a pair of single quotes ('') means it is a char constant. All the char constants are having a particular ASCII value. Operator '/'is having higher precedence than'-'. so, in the statement, '-'-'-'-'/'/'/' 1st the division will be performed and then the subtraction. Hence the output will be -1.

8. Ans: 1 1

Explanation : Value of x is 10, the statement x>9 is true and hence will return 1. Also x>3 and y!=3 both are true and hence value of q will be 1. So the output will be 1 1

9. Ans: Compilation error

Explanation : unsigned modifier cannot be used with float data type; it can only be used with the integral data type.

10. Ans: -5 2

Explanation : Here +i is a dummy statement. It has no effect in the given code.

11. Ans: Garbage value

Explanation : Here ptr is an integer pointer the address of variable a is stored in ptr after converting it to an integer type. *ptr is an integer value which is printed by using %f format specifier so the output is garbage value.

12. Ans: 12

Explanation : While comparison between the two values of different data type is done then the lower data type gets converted to the higher data type. So value of a gets converted to float and then the comparison takes place. So the condition inside the if statement becomes true and hence the output is 12.

13. Ans: -1 and 65535

Explanation : ~0 is the 1's complement of 0 which is nothing but 11111111 which is the binary representation of -1 as all the negative numbers are stored in the memory in the form of their 2's complements. So, the condition will be true and the output will be -1 and 65535.

14. Ans: Marathon

Explanation : In printf the format specifier %s requires the base address of a string in order to print it. Here we are giving "C Marathon"+2, so the address will get incremented by 2 bytes and now the address given to the printf function is the base address of M. so the output is Marathon.

15. Ans: equal

Explanation : ~ is the 1's complementing operator. So 1's complement of 0 is 11111111 which is the binary representation of -1 as all the negative numbers are stored in the form of their 2's complements. So the condition inside the if statement will be true and hence the output will be equal.

16. Ans: 145f

Explanation : In the first case "%d"+0 returns the address of string "%d" which will print 145. In the second printf "%f"+1 returns the address of "f ". Hence f gets printed and the final output is 145f.

---

17. Ans: Can't modify a constant object.

Explanation : Here p is a pointer which points to a constant object. So we cannot change the value of the object by using the pointer, but it can be changed using the name of the object. So, when we are writing ++(*p), it will give an error that cannot modify a constant object.

---

18. Ans: Compilation error

Explanation : The name of the enum type must be an identifier. Here there is a space between error and Type which is not an identifier. The 2nd problem is that we cannot give the same name to the enum variable and to its individual members

---

19. Ans: 5

Explanation : Here i is a constant object whose value cannot be changed. We are storing the address of i in a pointer p. then when we write *(p++), printf() will print value of *p(value at the address p i.e. 5) and then p will get incremented. So, the output will be 5.

---

20. Ans: 5

Explanation : Here i is a constant object and also p is a constant pointer which points to a const object. The address of i is assigned to p. then we are writing *(p++). So printf() will print value of *p (i.e. 5) and then p will get incremented.

---

21. Ans: Cannot modify a constant object.

Explanation : Here i is an int and p is a pointer to a constant object. (*p)++ means the value at address p is incremented, but here p is a pointer to a constant so compiler flags an error.

---

22. Ans: Declaration is not allowed here.

Explanation : All the declarations should be done at the first in the program, before writing the other statements. Here there are two null statements before the declaration of j. so the compiler will flag an error.

---

23. Ans: There is no problem in the given code.

Explanation : Here the declaration of i and j are done before multiple number of Null statements terminator. That's why it will not show any error

---

24. Ans: Size of void is unknown

Explanation : void is also known as generic data type. Its size is unknown. It is not known to the compiler that how much byte it should allocate for a. So, it will flag an error.

---

25. Ans: hai

Explanation : the escape sequence \n is for a new line char, \b shifts the control to the last char it printed, \r shifts the control to the beginning of the same line. Hence 1st ab will be printed then it is changed to asi and lastly changed to hai.

26. Ans: 2.

Explanation : We cannot declare two enum variables having same name of the individual member both inside the same scope. But, in a program, one is made global another is made local then, this can be done. As inside a scope, priority of local variable is higher than the global, so it will print 2

---

27. Ans: hi

Explanation : As 4.6 is recurring binary, and as the no of precision digits in float and double are different (in float it is up to 6 digits and in double it is up to 10 digits), although while comparison float will get converted to double, the value in the both will not the same. So the output will be hi.

---

28. Ans: Compilation error

Explanation : When we are writing something preceded by '\' and within a pair of single quotes ('\'), then it becomes an octal char const whose maximum range is 377.Hence it will show error as numeric constant too large

---

29. Ans: Numeric constant too large.

Explanation : enum datatype is used to create only a sequence of short integral constants whose maximum range is 32767. More than that it can't hold. Here if a is initialized with 32765, then value of d will be 32768 which is more than the range of short int. So the maximum possible value of a can be 32764, So that value of d can be 32767 which is the maximum range of short int.

---

30. Ans: printf("brainq");

Explanation : we know that printf() returns the no of character it prints on the screen. So, if we will write b=printf("brainq"); then 1st "brainq" will be printed and then printf() will return 6 which will get stored in b. So the output will be brainq25 6.

---

31. Ans:

1) enum col{BLACK,BLUE,GREEN};

2) #include<graphics.h>

3) int BLACK=0,BLUE=1,GREEN=2;

Explanation :

1) The enum data type creates a sequence of short integral constants and by default starting value is 0 for the first one. It can also be changed by providing a value to it.

2) In Turbo C in the header file graphics.h, BLACK, BLUE and GREEN are predefined constants with value 0, 1 and 2 respectively.

3) BLACK, BLUE and GREEN are declared as integer variable with values 0, 1 and 2.

---

32. Ans: 32767

Explanation : The loop will run for 32767 times. After that zero will occur and the condition inside while() will become false

33. Ans: x=x<<8 | x>>8;

Explanation : This statement will change the byte order of an integer. x is first shifted left 8 times and bitwise operated(OR) with x shifted right 8 times.

34. Ans: No output

Explanation : The decimal value of '\377' is 255. But in the for loop, the value of ch (ASCII value 48) is compared with '\377', 255 will be converted to signed value i.e. -1. So the condition in the for loop is not satisfied, hence no output.

35. Ans: Infinite loop

Explanation : As i is an int and its maximum range is 32767, which is always less than 40000(when it exceed 32767, enters to integer cycle), so results in infinite loop.

36. Ans: 10 0

Explanation : In Turbo C any number divided by 0, the result will be that number and in ANSI C there will be runtime error.

37. Ans:

```
void main()
{
  int x=300;
  if((*((unsigned char*)&x)==1)&&(*((unsigned char*)&x+1)==44))
     printf("BIG ENDIAN");
  else
     printf("LITTLE ENDIAN");
}
```

38. Ans:

```
#define any_size(any) (char*)(&any)-(char*)((&any)-1)
void main()
{
  //any type of variable like int c;
  printf("%d",any_size(c));
}
```

39. Ans: Declaration syntax error

Explanation : To debug the program in ganga.h the statement extern int x++; should be changed to extern int x; The statement extern int x; tells the compiler that the variable x is defined somewhere else in the program or may be in other file.

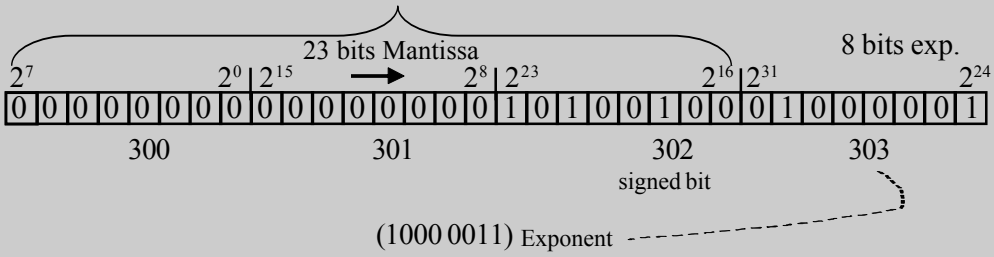
40. Ans:

float x= 20.5 ;

The binary equivalent of 20 = 10100 and of 0.5 = .1

That means, 20.5 = 10100.1

In normalized form, it will be **1.01001** * e^4.



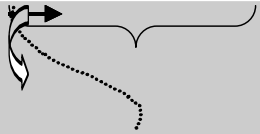
According to the above example, the exponent 4 is added with bias 127 and stored in exponent 8 bits.

$$\begin{array}{r} 127 \\ +\ 4 \\ \hline 131 \text{ - } 10000011 \end{array}$$

41. Ans: int f(int,float);

Explanation : If the function is called before its execution its prototype is necessary before the function call.

42. Ans:



```
#include"limits.h"
void main()
{
  if(INT_MAX+1==INT_MIN)
     printf("CYCLE IS PRESEN");
  else
     printf("CYCLE IS NOT PRESENT");
}
```

43. Ans:

```
void main()
{
  int n;
  int c=0;
  printf("\nEnter any number:");
  scanf("%d",&n);
  while(x>=1)
  {
     c++;
     n=n&n-1;
  }
  printf("%d",c);
}
```

44. Ans: 20

Explanation : int x is a auto variable which creates and initialized at runtime. So at the runtime it will initialized with the return value of function get() which is 20.

45. Ans:

```
void main()
{
  int n;
  int c=0,i=0;
  printf("\nEnter any number:");
  scanf("%d",&n);
  while(i<=15)
  {
     if(n&1==1)
        c++;
     n=n>>1;
     i++;
  }
  printf("%d",c);
}
```

46. Ans: Line no. 9

Explanation : We can't dereference a void type of pointer.

47. Ans: 22

Explanation : 300 is first typecasted to char data type that will give 44 (due to the presence of cycle in char data type) and then divided by 2 which gives 22.

48. Ans: -32536

Explanation : The maximum limit of the integer variable is 32767. We are assigning 33000 to x which results in overflow and due to the presence of cycle in integer datatype 33000 wraps to give -32536.

49. Ans: Hi

Explanation : By default the floating point constant is double. As 4.3 is a recurring binary the memory representation of both x and 4.3 will differ. So, the if condition is false and the output will be Hi.

50. Ans: 1000

Explanation : 0x1234's memory representation is 0001 0010 0011 0100. When x is shifted 12 bits right we get 0000 0000 0000 0001. Again x is shifted 12 bits left now we get 0001 0000 0000 0000. That equals to 0x1000.