2 VARIABLES

This chapter takes you around the nook and corner of variables, data types, and its values. After going through this chapter you will be able to select and use appropriate variables and their characteristics.

1. What is the output of the code? (L1)

```
void main()
{ int a=8;
int b=a;
printf("%d %d",a,b); }
```

(a) 8 8 (b) 8 0
(c) Error (d) 8 garbage_value

Answer: a) 'a' is declared first and assigned to 'b' later.

2. What is the output of the code? (L2)

```
void main()
{ int a=8;
int b=c=a;
printf("%d %d %d",a,b,c); }
```

(a) 8 8 8
(b) 8 0 0
(c) Error
(d) 8 garbage_value garbage_value

Answer: c) 'a' is declared first. The second declares 'b' and assigns the value of 'a' to 'c' and from there to 'b', but 'c' is never declared hence throws error.

3. What is the output of the code? (L1)

```
void main()
{ int a=8;
int b=a=c;
printf("%d %d %d",a,b,c); }
```

(a) 8 8 8
(b) 0 0 0
(c) Error
(d) garbage_value garbage_value garbage_value

Answer: c) 'a' is declared first. The second declares 'b' and assigns the value of 'c' to 'a' and from there to 'b', but 'c' is never declared hence throws error.

4. What is the output of the code? (L3)

```
void main()
{ int a=8;
int b=a=4;
printf("%d %d",a,b); }
```

(a) 8 8 (b) 8 0
(c) 4 4 (d) 4 0

Answer: c) The second value of 'a' overwrites the first value and the same is assigned to 'b'.

5. What is the output of the code? (L2)

```
void main()
{ int a=8;
int b=a,a=4;
printf("%d %d",a,b); }
```

(a) 8 8 (b) 8 4
(c) 4 4 (d) Error

Answer: d) Re-declaration of a variable within same scope is not allowed.

Answer: d) The float value matches itself to the specifier and the undefined specifer takes the garbage value.

13. What is the output of the code? (L3)

```
void main()
{ int a;
float b;
printf("%f %d",a=3,b=5.14); }
```

(a) 0 garbage_value
(b) 0 0
(c) 5.140000 3
(d) 3.000000 5

Answer: c) In spite of wrong order the arguments matches to the specifiers by themselves.

14. What is the output of the code? (L3)

```
void main()
{ float b=0.23;
printf("%f %f",b=5.14,b); }
```

(a) 5.140000 5.140000
(b) 5.140000 .230000
(c) Garbage_value Garbage_value
(d) 0.230000 0.230000

Answer: b) It just executes from right to left.

15. What is the output of the code? (L2)

```
void main()
{ char d;
printf("%c",d); }
```

(a) Garbage_value (b) Blank space
(c) `\0` (d) None of the above

Answer: d) By default all `char` variables are initiated with null but on printing nothing will be printed and the compiler terminates. All other data types initiate to 0.

16. What is the output of the code? (L2)

```
void main()
{ oct d;
printf("%o",d=9); }
```

(a) Garbage_value (b) 9
(c) 11 (d) Error

Answer: d) `'oct'` is not a valid data type.

17. What is the output of the code? (L2)

```
void main()
{ string d;
printf("%s",d="hai"); }
```

(a) Garbage_value (b) `hai`
(c) Blank space (d) Error

Answer: d) String is not a valid data type.

18. What is the output of the code? (L2)

```
void main()
{ int g;
printf("%x",g); }
```

(a) Garbage_value (b) 0
(c) 0x00 (d) Error

Answer: c) Even though the format specifer is `%x` the intial value 0 is not printed in hexadecimal format.

19. What is the output of the code? (L2)

```
void main()
{ float g;
printf("%x",g); }
```

(a) Garbage_value (b) 0
(c) 0x00 (d) Error

Answer: a) The initial value 0.000000 is a float and does not have a hexadecimal value. The internal difference in the interpretation of the formats results in some garbage values.

20. What is the output of the code? (L2)

```
void main()
{ const float g=3.3;
printf("%f",g*g); }
```

(a) Garbage_value (b) Logical error
(c) 10.88999 (d) Error

Answer: c) `printf` doesn't assign but just calculates the value hence not an error.

21. What is the output of the code? (L1)

```
void main()
{ const float g=3.3;
printf("%f",g=g*g); }
```

(a) Garbage_value (b) Logical error
(c) 10.88999 (d) Error

Answer: d) `const` variable cannot be edited.

22. What is the output of the code? (L2)

```
void main()
{ const float g;
printf("%f",g*g); }
```

(a) Garbage_value (b) Logical error
(c) 0.000000 (d) Error

Answer: c) Since the value of 'g' is not assigned the value it is not an error.

23. What is the output of the code? (L2)

```
void main()
{ const float g;
printf("%f",g=g*g); }
```

(a) Garbage_value (b) Logical error
(c) 0.000000 (d) Error

Answer: d) 'const' variable cannot be edited.

24. What is the output of the code? (L2)

```
void main()
{ long float g=2.3431;
printf("%f",g); }
```

(a) Garbage_value (b) Logical error
(c) 2.3431000000 (d) Error

Answer: d) Long cannot be used with float.

25. What is the output of the code? (L2)

```
void main()
{ long int g=1234;
printf("%d",g); }
```

(a) Garbage_value (b) Logical error
(c) 1234 (d) Error

Answer: c) %d does not support long int, as the value is small it prints the same.

26. What is the output of the code? (L3)

```
void main()
{ int v=1234509865;
printf("%d",v); }
```

(a) Garbage_value
(b) Logical error
(c) 1234509865
(d) Compiler dependent

Answer: d) The range of integer is different in various compilers. In some compilers this may be supported and prints without error. For example 'gcc' supports 9 digits.

27. What is the output of the code? (L3)

```
void main()
{ short int v=12345;
printf("%d",v); }
```

(a) Compilation error
(b) Logical error
(c) 12345
(d) Compiler dependent

Answer: d) The range of short modifier is different in various compilers. In some compilers this may be supported and prints without error. For example 'gcc' supports 4 digits without any modifications.

28. What is the output of the code? (L2)

```
void main()
{ double v=1.234527938;
printf("%lf",v); }
```

(a) Compilation error
(b) Logical error
(c) 1.234527938
(d) 1.234528

Answer: d) The value is stored without any problem and the range is based on the compiler. While displaying the precision is same for float and double.

29. What is the output of the code? (L2)

```
void main()
{ unsigned v= -8;
printf("%u",v); }
```

(a) Error- illegal data type
(b) Biased value of -8
(c) Error- illegal data
(d) -8

Answer: b) Unsigned has a fixed range and the bias value, internally the interpretation is different. Even a negative value internally gets its biased value.

36. What is the output of the code? (L2)

```
void main()
{ signed int g= 3124566;
printf("%d",g); }
```

(a) -3124566
(b) +3124566
(c) Error — multiple specifiers in same declaration
(d) 3124566

Answer: d) Signed modifier is just like default `int` and prints the same output. It is to be noted that it will not print '+'.

37. What is the output of the code? (L2)

```
void main()
{ signed float h= -31.96;
printf("%f",h); }
```

(a) -31.96
(b) +31.96
(c) Error — multiple specifiers in same declaration
(d) 31.96

Answer: c) Signed modifier is compatible only with integer.

38. What is the output of the code? (L2)

```
void main()
{ short float h= -31.9676586;
printf("%f",h); }
```

(a) -31.96
(b) +31.96
(c) Error — multiple specifiers in same declaration
(d) 31.96

Answer: c) Short modifier is compatible only with integer.

39. What is the output of the code? (L2)

```
void main()
{ const float h= -3.2;
printf("%f",h*h); }
```

(a) 10.240000
(b) -3.2000000
(c) Error — multiple specifiers in same declaration
(d) Garbage_value

Answer: a) Constant is not updated, only calculated and printed. So there is no error.

40. What is the output of the code? (L3)

```
void main()
{ const char h= 'f';
printf("%c",h+h); }
```

(a) f (b) f+f
(c) f f (d) None of the above

Answer: d) The operations calculated the `ascii` sum of `'h+h'` and then tries to convert it to `char`, but as there is no corresponding `char` it just terminates the program without printing anything.

41. What is the output of the code? (L3)

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

(a) \n (b) n
(c) New line (d) Segmentation fault

Answer: c) The output just prints a new line. Escape sequence are treated as single characters and can be stored in a single char variable.

42. What is the output of the code? (L3)

```
void main()
{ char h= '\t';
printf("%c",h); }
```

(a) \t
(b) t
(c) Horizontal space
(d) Segmentation fault

Answer: c) The output just prints a horizontal space.

43. What is the output of the code? (L3)

```
void main()
{ char h= '\n\t';
printf("%c",h); }
```

(a) Horizontal space alone
(b) Newline alone
(c) Newline and a horizontal space
(d) Segmentation fault

Answer: a) There is internal overflow of constant conversion and `\t` overwrites `\n`.

44. What is the output of the code? (L2)

```
void main()
{ char h= 'nt';
printf("%c",h); }
```

(a) `t` (b) `nt`
(c) Segmentation fault

Answer: a) There is internal overflow of constant conversion and `t` overwrites `n`.

45. What is the output of the code? (L3)

```
void main()
{ char h= "nt";
printf("%c",h); }
```

(a) `n` (b) `t`
(c) `nt` (d) None of the above

Answer: d) As the values are not given in single quotes their integer values are not recognised by default and hence prints nothing.

46. What is the output of the code? (L3)

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

(a) `d` (b) `%d`
(c) Error (d) None of the above

Answer: d) As the values are not given in single quotes their integer values are not recognised by default and hence prints nothing.

47. What is the output of the code? (L3)

```
void main()
{ int a=3;
char h= "%d";
printf(h,a); }
```

(a) `3`
(b) `%d`
(c) Computational error
(d) None of the above

Answer: d) `'h'` is a variable and results in segmentation fault.

48. What is the output of the code? (L2)

```
void main()
{ char h= '%d';
printf("%c",h); }
```

(a) `d`
(b) `%d`
(c) Computational error
(d) None of the above

Answer: a) `'%'` and `'d'` are treated as separate characters and `'d'` overwrites `'%'` and hence prints `'d'`.

49. What is the output of the code? (L2)

```
void main()
{ int a=3;
char h= '\t',g='\n';
printf("%c%d%c",h,a,g); }
```

(a) Tab followed by 3 and takes the control to next line
(b) Tab followed by 3 and terminates
(c) Computational error
(d) None of the above

Answer: a) The escape characters stored inside the char variables work normally and hence print usual output.

50. What is the output of the code? (L2)

```
void main()
{ int a=3;
char h= '\t',g='\n';
printf("%c%d%c",h,g,a); }
```

(a) Tab followed by 3 and takes the control to next line
(b) Tab followed by 3 and terminates
(c) Computational error
(d) None of the above

Answer: d) The mismatch of the format specifiers and the variables cause data type conversion and it prints a computational garbage.

51. What is the output of the code? (L2)

```
void main()
{ int a=3;
char h= '%',g='d';
printf(hg,a); }
```

(a) 3
(b) Logical Error
(c) Computational error
(d) None of the above

Answer: b) Here the `'hg'` is treated as a single variable and leads to compilation error.

52. What is the output of the code? (L3)

```
void main()
{ char h= '\r';
printf("hello%chai",h); }
```

(a) `hello` (b) `hai`
(c) `hellohai` (d) `hello\rhai`

Answer: b) The `\r` stored inside the `char` variable works as normal escape character carriage return.

53. How many spaces does a `"\t"` escape character prints in the console? (L3)

(a) 3
(b) 5
(c) 1
(d) Compiler dependent

Answer: a) Tab is 3 spaces.

54. What is the output of the code? (L3)

```
void main()
{ char h= '\b';
printf("Hello\t%cworld",h); }
```

(a) `Hello world`
(b) `Hello world`
(c) `Helloworld`
(d) `Hello\world`

Answer: b) It is based on the fact that `\t` prints 3 spaces. `'\b'` deletes one space, so remaining two will be printed.

55. What is the output of the code? (L1)

```
void main()
{ char h= '\n';
printf("%-c",h); }
```

(a) newline (b) `-c\n`
(c) `-cn` (d) `cn`

Answer: a) The `'-'` in the specifier is neglected as usual.

56. What is the output of the code? (L3)

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

(a) newline (b) `-c\n`
(c) `-cn` (d) `n`

Answer: d) Here space `(' ')` overwrites `'\'` and `'n'` overwrites it in turn.

57. What is the ASCII value of `'\n'`? (L2)

(a) 8 (b) 10
(c) 7 (d) 21

Answer: b)

58. What is the ASCII value of `'\f'`? (L2)

(a) 8 (b) 9
(c) 12 (d) 13

Answer: c)

59. What is the ASCII value of `'\t'`? (L2)

(a) 9 (b) 12
(c) 11 (d) 45

Answer: a)

60. What is the ASCII value of `'\a'`? (L2)

(a) 14 (b) 8
(c) 7 (d) 23

Answer: c)

61. What is the ASCII value of `'\r'`? (L2)

(a) 11 (b) 15
(c) 13 (d) 3

Answer: c)

79. What is the output of the code? (L2)

```
void main()
{ float int z= 9.00;
printf("%d",z); }
```

(a) 9
(b) 9.00000
(c) Compilation error
(d) Logical error

Answer: c) Multiple data types cannot be specified for same variable.

80. What is the error in the code? (L3)

```
void main()
{ float a;
int f= 7;
printf("%d %f",a,a); }
```

(a) Mismatch of arguments in `printf` statement
(b) Multiple declaration of a same variable `'a'`.
(c) Conflicting types for variable `'a'`.
(d) Logical error

Answer: c) In spite of multiple declaration the data type is given more importance.

81. What is the output of the code? (L2)

```
void main()
{ const int f;
int f= 7;
printf("%d ",f); }
```

(a) Compilation error
(b) 7
(c) Unpredictable behaviour
(d) Logical error

Answer: a) Same variable cannot be re-declared.

82. What is the output of the code? (L2)

```
void main()
{ const float f=3;
printf("%f ",f); }
```

(a) Compilation error
(b) 3
(c) 3.000000
(d) Logical error

Answer: c) The `const` keyword does violate the basic properties of internal conversion. It computes the best answer that is possible.

83. What is the output of the code? (L2)

```
void main()
{ const int f=3.1412;
printf("%d ",f); }
```

(a) Compilation error
(b) 3
(c) 3.000000
(d) Logical error

Answer: b) The `const` keyword does violate the basic properties of internal conversion. It computes the best answer that is possible.

84. What is the output of the code? (L2)

```
void main()
{ near int f;
printf("%d ",f); }
```

(a) Compilation error
(b) 0
(c) Garbage value
(d) Logical error

Answer: a) `near` is not an integer modifier. So undefined type near.

85. What is the output of the code? (L2)

```
void main()
{ far int f;
printf("%d ",f); }
```

(a) 0 (b) Compilation error
(c) Garbage value (d) Logical error

Answer: b) `far` is not an integer modifier. So undefined type `far`. It also make the sentence incomplete.

86. What is the output of the code? (L1)

```
void main()
{ char *;
printf("%c ",*); }
```

94. What is the error in the code? (L3)

```
void main()
{ char ;
printf("%c",a); }
```

(a) Useless type defined char
(b) Undeclared variable 'a'
(c) No error
(d) Logical error

Answer: b) The data type without any argument is warned as useless but doesn't bother the compiler.

95. What is the output of the code? (L3)

```
void main()
{ char ;
printf("this is a sample program"); }
```

(a) Prints nothing
(b) Compilation error
(c) This is a sample program
(d) Logical error

Answer: c) The data type without any argument is warned as useless but doesn't bother the compiler.

96. What is the output of the code in turbo c 3.5 compiler? (L2)

```
void main()
{ char aabbccddeeffgghhiijjkkllmmnnooppqqrrss= 's';
char aabbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvv= 'c'
printf("%c ", aabbccddeeffgghhiijjkkllmmnnooppqqrrss); }
```

(a) Prints nothing (b) Compilation error
(c) c (d) s

Answer: b) The variable name is significant only for 32 characters and after that it will not be considered. So here the compiler throws multiple declaration error. It is to be noted that gcc compiler doesn't has that constrain.

97. In a 64-bit processor with gcc compiler what is the output of the program? (L1)

```
void main()
{ char a='h';
printf("%d",sizeof(a)); }
note*-sizeof() is an predefined
function that gives the size
of the argument given as input.
```

(a) 4
(b) 1
(c) Compilation error
(d) 2

Answer: b)

98. In a 64-bit processor with gcc compiler what is the output of the program? (L1)

```
void main()
{ int a=54313;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 1
(c) Compilation error
(d) 2

Answer: a)

99. In a 64-bit processor with gcc compiler what is the output of the program? (L1)

```
void main()
{ float a=543.13;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2

Answer: a)

100. In a 64-bit processor with gcc compiler what is the output of the program? (L1)

```
void main()
{ double a=543.13;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8

(c) Compilation error
(d) 16
Answer: b)

101. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ long a=5431343563;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 16
Answer: b)

102. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ short int a=54;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2
Answer: d)

103. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ unsigned int a=4;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2
Answer: a)

104. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ volatile int a;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2
Answer: a)

105. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ volatile a;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2
Answer: a)

106. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ volatile double a;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 8
(c) Compilation error
(d) 2
Answer: b) Volatile takes the size of double leaving the original identity.

107. In a 64-bit processor with `gcc` compiler what is the output of the program? (L1)

```
void main()
{ volatile char a;
printf("%d",sizeof(a)); }
```

(a) 4
(b) 1
(c) Compilation error
(d) 2
Answer: b) Volatile takes the size of `char` leaving the original identity.

(a) 4 bytes (b) 0 bytes
(c) 1 bytes (d) 8 bytes

Answer: b) Memory is allocated only to variables. So useless data type doesn't consume memory.

115. What is the output of the program? (L4)

```
void main()
{ printf("%d",int a=10); }
```

(a) 10 (b) Error
(c) 0 (d) Prints nothing

Answer: b) Declaration cannot be inside any 'printf' arguments.