Data Type and Size  
**Writer:** Sayak Haldar

**1. Comment on the output of this C code?**

#include <stdio.h>

int main()

{

int a[5] = {1, 2, 3, 4, 5};

int i;

for (i = 0; i < 5; i++)

if ((char)a[i] == '5')

printf("%d\n", a[i]);

else

printf("FAIL\n");

}

a) The compiler will flag an error

b) Program will compile and print the output 5

c) Program will compile and print the ASCII value of 5

d) Program will compile and print FAIL for 5 times

**Answer: d**

Explanation: The ASCII value of 5 is 53, the char type-casted integral value 5 is 5 only.

Output:

$ cc pgm1.c

$ a.out

FAILED

FAILED

FAILED

FAILED

FAILED

**2. The format identifier ‘%i’ is also used for \_\_\_\_\_ data type?**

a) char

b) int

c) float

d) double

**Answer: b**

Explanation: Both %d and %i can be used as a format identifier for int data type.

**3. Which data type is most suitable for storing a number 65000 in a 32-bit system?**

a) signed short

b) unsigned short

c) long

d) int

**Answer: b**

Explanation: 65000 comes in the range of short (16-bit) which occupies the least memory. Signed short ranges from -32768 to 32767 and hence we should use unsigned short.

**4. Which of the following is a User-defined data type?**

a) typedef int Boolean;

b) typedef enum {Mon, Tue, Wed, Thu, Fri} Workdays;

c) struct {char name[10], int age};

d) all of the mentioned

**Answer: d**

Explanation: typedef and struct are used to define user-defined data types.

**5. What is the size of an int data type?**

a) 4 Bytes

b) 8 Bytes

c) Depends on the system/compiler

d) Cannot be determined

**Answer: c**

Explanation: The size of the data types depend on the system.

**6. What is the output of this C code?**

#include <stdio.h>

int main()

{

signed char chr;

chr = 128;

printf("%d\n", chr);

return 0;

}

a) 128

b) -128

c) Depends on the compiler

d) None of the mentioned

**Answer: b**

Explanation: signed char will be a negative number.

Output:

$ cc pgm2.c

$ a.out

-128

**7. Comment on the output of this C code?**

#include <stdio.h>

int main()

{

char c;

int i = 0;

FILE \*file;

file = fopen("test.txt", "w+");

fprintf(file, "%c", 'a');

fprintf(file, "%c", -1);

fprintf(file, "%c", 'b');

fclose(file);

file = fopen("test.txt", "r");

while ((c = fgetc(file)) != -1)

printf("%c", c);

return 0;

}

a) a

b) Infinite loop

c) Depends on what fgetc returns

d) Depends on the compiler

**Answer: a**

**Now, what does w+ mode mean?**Create a text file for reading and writing. If the file exists, it is overwritten.

Output:

$ cc pgm3.c

$ a.out

a

**8. What is short int in C programming?**

a) Basic datatype of C

b) Qualifier

c) Short is the modifier and int is the basic datatype

d) All of the mentioned

**Answer: c**

**What is type qualifier?**In the C, C++, and D programming languages, a type qualifier is a keyword that is applied to a type, resulting in a qualified type. For example, const int is a qualified type representing a constant integer, while int is the corresponding unqualified type, simply an integer.

There are two types of qualifiers available in C language. They are,

const

Volatile

**1. CONST KEYWORD:**

Constants are also like normal variables. But, only difference is, their values can’t be modified by the program once they are defined.

They refer to fixed values. They are also called as literals.

They may be belonging to any of the data type.

**Syntax:**

const data\_type variable\_name; (or) const data\_type \*variable\_name;

**2. VOLATILE KEYWORD:**

When a variable is defined as volatile, the program may not change the value of the variable explicitly.

But, these variable values might keep on changing without any explicit assignment by the program. These types of qualifiers are called volatile.

For example, if global variable’s address is passed to clock routine of the operating system to store the system time, the value in this address keep on changing without any assignment by the program. These variables are named as volatile variable.

**Syntax:**

volatile data\_type variable\_name; (or) volatile data\_type \*variable\_name;

**Modifiers In c Language:**The amount of memory space to be allocated for a variable is derived by modifiers.

Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable.

For example, storage space for int data type is 4 byte for 32 bit processor. We can increase the range by using long int which is 8 byte. We can decrease the range by using short int which is 2 byte.

**There are 5 modifiers available in C language. They are**

short

long

signed

unsigned

long long

**9. Comment on the output of this C code?**

#include <stdio.h>

int main()

{

float f1 = 0.1;

if (f1 == 0.1)

printf("equal\n");

else

printf("not equal\n");

}

a) equal

b) not equal

c) output depends on compiler

d) none of the mentioned

**Answer: b**

Explanation: 0.1 by default is of type double which has different representation than float resulting in inequality even after conversion.

Output:

$ cc pgm4.c

$ a.out

not equal

**10. Comment on the output of this C code?**

#include <stdio.h>

int main()

{

float f1 = 0.1;

if (f1 == 0.1f)

printf("equal\n");

else

printf("not equal\n");

}

a) equal

b) not equal

c) output depends on compiler

d) none of the mentioned

**Answer: a**

Explanation: 0.1f results in 0.1 to be stored in floating point representations.

Output:

$ cc pgm5.c

$ a.out

equal

**11. What is the output of this C code (on a 32-bit machine)?**

#include <stdio.h>

int main()

{

int x = 10000;

double y = 56;

int \*p = &x;

double \*q = &y;

printf("p and q are %d and %d", sizeof(p), sizeof(q));

return 0;

}

a) p and q are 4 and 4

b) p and q are 4 and 8

c) Compiler error

d) p and q are 2 and 8

**Answer: a**

Explanation: Size of any type of pointer is 4 on a 32-bit machine.

Output:

$ cc pgm6.c

$ a.out

p and q are 4 and 4

**12. Which is correct with respect to size of the datatypes?**

a) char > int > float

b) int > char > float

c) char < int < double

d) double > char > int

**Answer: c**

Explanation: char has lesser bytes than int and int has lesser bytes than double in any system

**13.What is the output of the following C code(on a 64 bit machine)?**

#include <stdio.h>

union Sti

{

int nu;

char m;

};

int main()

{

union Sti s;

printf("%d", sizeof(s));

return 0;

}

a) 8

b) 5

c) 9

d) 4

**Answer: d**

Explanation: Since the size of a union is the size of its maximum datatype, here int is the largest hence 4.

Output:

$ cc pgm7.c

$ a.out

4

**14. What is the output of this C code?**

#include <stdio.h>

int main()

{

float x = 'a';

printf("%f", x);

return 0;

}

a) a

b) run time error

c) a.0000000

d) 97.000000

**Answer: d**

Explanation: Since the ASCII value of a is 97, the same is assigned to the float variable and printed.

Output:

$ cc pgm8.c

$ a.out

97.000000

**15. Which of the datatypes have size that is variable?**

a) int

b) struct

c) float

d) double

**Answer: b**

Explanation: Since the size of the structure depends on its fields, it has a variable size.