Tutorial Link https://codequotient.com/tutorials/Arrays and Memory/59fde6f5e63d6b7fd5dec01e

TUTORIAL

Arrays and Memory

Chapter

1. Arrays and Memory

The amount of memory required for an array is directly related to its type and size. For a one dimension

array, the total size in bytes can be computed:

```
total_bytes = sizeof(one_element) × size_of_array
```

Also C has no bounds checking on arrays. So programmer can write more number of elements and compiler will never warn, instead it will overwrite the memory elements at consecutive locations. As the programmer, it is your job to provide bounds checking where needed. For example, this code will compile without error, but it is incorrect because the loop will cause the array to be overrun: -

C

```c
#include <stdio.h>
int main()
{

  int age[5], i;
  for(i=0; i<10; i++)
    age[i] = i;

  for(i=0; i<10; i++)
    printf("%d\n",age[i]);
```

```
11
12      return 0;
13  }
```

In memory say the array age will start storing at address 1000 and each element will take 4 bytes then the memory map after this program will be: -

Values	0	1	2	3	4	5	6	7	8	9
Variables	Age[0]	Age[1]	Age[2]	Age[3]	Age[4]					
Address	1000	1004	1008	1012	1016	1020	1024	1028	1032	1036

So the code above will automatically save the 10 values at 10 locations (from 1000 to 1036) and will overwrite the values of other parts of memory not belonging to array (other variables, other sections of program etc.) which will make some chaotic situations and some unpredictable behaviors of the program. So you have to take care of it as we are habitual for counting from 1 and hence sometimes we may start the loop from 1 up to 5, in that case the program will run smoothly but may behave in some undesirable behavior.