Programming fundamentals

We studied that size of struct is computed by adding up all primitive data types used

Person bill

```
struct Person
{
    char c;
    int age;
};
```

memory of int is 4 bytes

memory of char is 1 byte

5 bytes of memory is allocated for structure variable bill

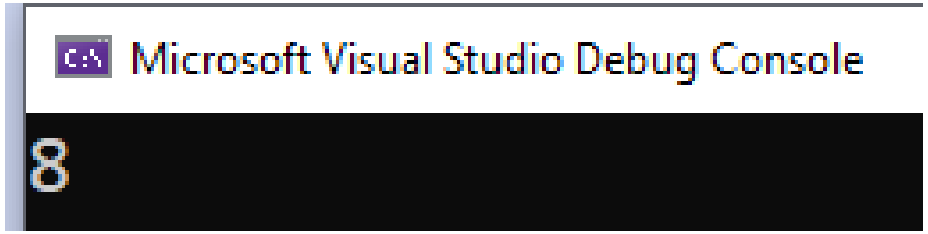
But if you run code and find size using sizeof function then the answer will be slightly different

```cpp
#include <iostream>
using namespace std;

struct Person
{
    char c;
    int age;
};

int main()
{
    Person p;
    cout << sizeof(p);
return 0;
}
```

It should be 5! But the reason that why it is 8 is that compiler uses extra bytes (8-5) in doing memory management (padding)



What is padding?

- Padding means to consider chunks of 4 bytes (in case of 32 but system) everytime while accessing anything
- That means if we came across char variable and we know it consumes 1 byte..but our system will access it in 1 word (4 bytes)
- Therefore, the compiler will use 1 byte in storing char and 3 bytes taken as extra bytes (padded at the end to make the chunk of 4 byte)

```
struct Person
{
    char c;
    int age;
};
```

8 bytes of memory is allocated for structure variable bill

8 bytes		
c	Padded bytes	age
1 byte	3 bytes	4 bytes

Note

- If the struct has only one variable of one primitive type..then there is no need for padding..
- Padding usually applies when there are multiple variables
- In summary, extra bytes are consumed in organizing the memory (padding)