Let's say you have a problem, you want to print the average score for each subject for someone throughout their 3 exams in the year.

You have say 10 subjects that you have to accommodate:

1. Mathematics
2. Computer Sc.
3. Physics
4. Chemistry
5. EVS
6. SUPW
7. English Language
8. English Literature
9. Hindi Language
10. Hindi Literature

Let's say that, the subjects are addressed by subject code, which is 1 for mathematics, 2 for CSc, 3 for Physics and so on.

How do you write this program?

Some of you may go:

> Well you start by declaring 30 variables, 3 per subject, then you input the marks of each subject in each exam in their respective variable, then you print their averages by adding them and dividing by 3.

Some of you may try and reduce the declarations like so:

> Instead of declaring 30 variables, declare 10, then create a temporary variable to take input and add the value of this variable to the required variable, finally divide it by 3 to get your average.

But both of those approaches require a lot of duplication of code which works for small numbers like 10, but what if the problem was a little different and the number was larger like a hundred or a thousand or even ten thousand for that matter say. You can't work with 100+, 1000+ or 10000+ variables, you won't even remember their names at that point.

So the Computer scientist came up with an ingenious solution to this.

They made a keen observation, you see, they realized that, since you need to store say n integers, you already know that every integer takes 4 bytes right, so just create a memory region of size 32 * n bits.

Then assign the first 4 bytes to first integer, the next 4 bytes to second, so on to the last 4 bytes to the nth integer.

This way, you can access the nth integer, by simply shifting it 4 * (n - 1) bytes forward from the starting position.

In general for a type of size s, you can shift the address s * (n - 1) bits and you'd be good to go.

This shifting of pointer to right or left is called pointer arithmetic.

Now here comes the best part, since arrays can only be homogenous, i.e. they can only contain data of 1 data type, and C knows the size of all the data types, you only need to state how many places you wanna jump forward, C fills in the size of the type itself.

> Ok, that's cool and all but like how do we use arrays?

Arrays declaration syntax in C, looks like so:

```
<Type> <array name>[<number of elements>];
```

So to declare an array of 10 integers called `my_arr`, you'd write

```c
int my_arr[10];
```

To initialize array with elements, you can put the elements in braces, separated by commas on the right side of equals to sign then end the statement with a semicolon.

For ex:

```c
int my_arr[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

Note, if you're using `gcc` or the gnu c compiler, you can write the following and it will work just fine.

```c
int my_arr[10] = { [0 ... 9] = 0};
```

If you're not using `gcc` though, you can also just conjure your own for loop to do this for you.

```c
int my_arr[10];
for(int i = 0; i < 10; ++i) {
    *(my_arr + i) = 0;
}
```

`my_arr + i` shifts the pointer `i * 32` bits, since size of `int` is 4 bytes (well on most modern computers it is)

Now, since accessing ith position is an common operation, C developers decided, they should create a better way to express this, so they came up with the [] operator for accessing elements, also called the array subscript operator.

Basically instead of writing `*(my_arr + i)` you can write `my_arr[i]` and it will act in the same manner, in fact it will be converted to the first expression before compilation.

> **Fun Titbit**
>
> What C developers didn't account for(or accounted for but chose to ignore it), was that since addition is commutative, and `x[i]` de-compiles to `*(x + i)` you can also write `i[x]` and that would work totally fine.

Now, with our freshly acquired knowledge of arrays, let's have a try at our problem.

```c
#include <stdio.h>

int main(int argc, char **argv) {
    double marks[10];

    for(int i = 0; i < 10; ++i) {
        printf("Enter the marks obtained in subject code %d, seperated by single space", (i + 1));
        double marks_sem_1 = 0.0;
        double marks_sem_2 = 0.0;
        double marks_sem_3 = 0.0;

        scanf("%lf %lf %lf", &marks_sem_1, &marks_sem_2, &marks_sem_3);

        marks[i] = (marks_sem_1 + marks_sem_2 + marks_sem_3) / 3.0;
    }

    for(int i = 0; i < 10; ++i) {
        printf("Your average marks for %d subject, are %f", i + 1, i[marks]);
    }
}
```

If you have trouble understating what is happening, try a hand unrolling of the loop and it should be crystal clear how this works.

[[Exploring Arrays]]