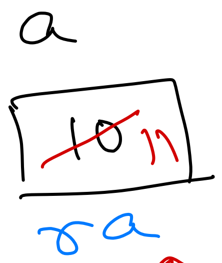
Reference

```
int main() {
    int a = 10;
    int & ra = a;
    ++ra;
    int & r;
```



`&` → define a reference variable.

`int & ra = a;` → Once a reference is initialized, it can not be changed to refer to some other variable.

`int & r;` ✗ → Reference must always be initialized.

```
const int x = 10;

int & rx = x; ✗
```

$\underbrace{\texttt{int \&}}_{\text{non- constant reference}}$ rx = $\underset{\text{constant}}{x}$;

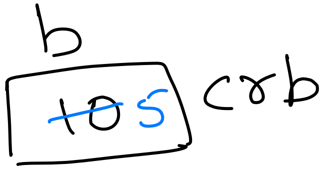
Can not have non- constant ref. to constant.

```
const int & crx = x; ✓
```

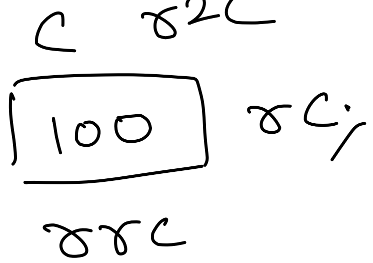
```
int b = 10;

const int & crb = b;

b = 5;

crb = 100; ✗
```

$\underbrace{\texttt{const int}}_{\text{ref to constant value.}}$

→ Hence can't change it.



```
int   c = 100;

int &rc = c;

int & rrc = rc;

int & r2c = c;
```



Arrays → a variable in which we can store multiple values of same type.

int x, y, z;

size of array (max number of elements that can be stored) is decided at compile time.

data_type variable name [size] ;

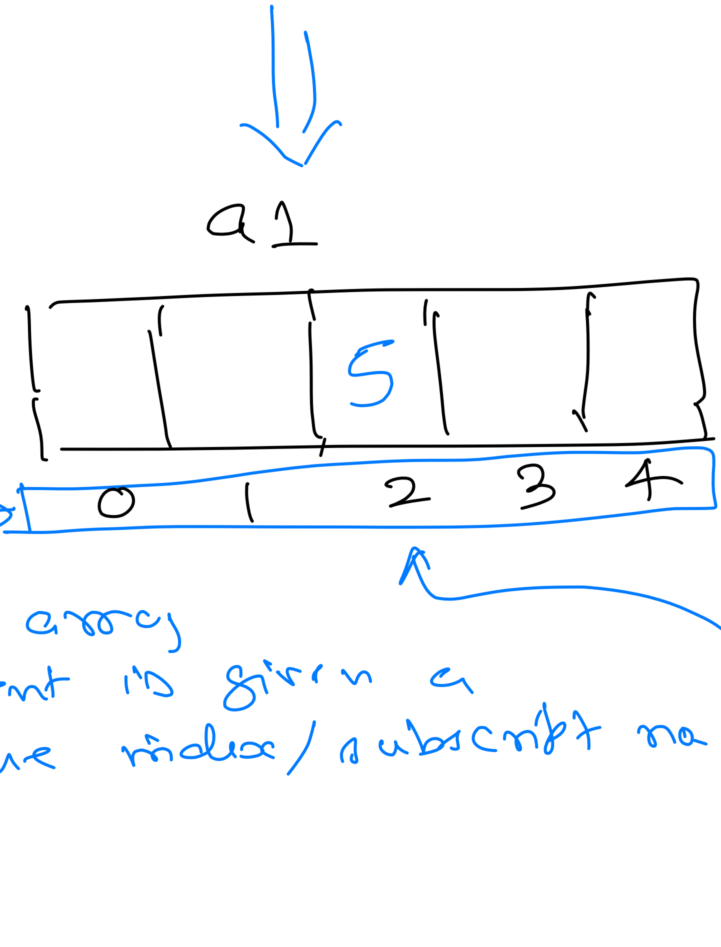
size is mandatory

data_type ↑ type of each element of array.

variable name ↑ array name

size ↑ max number of elements in array

int a1[5]; → a1 is an array of 5 elements, each of type int.



memory for array is allocated in a single memory block

each array element is given a unique index / subscript no

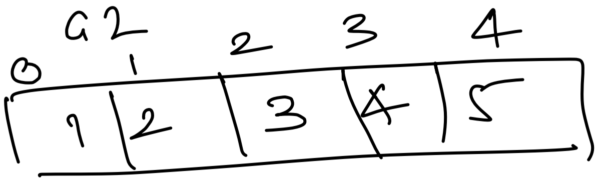
$\boxed{a[2]} = 5;$

array subscript operator

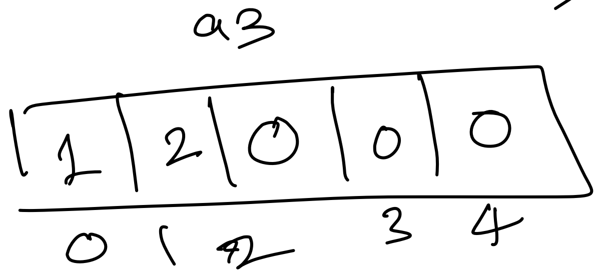
Can be used at a place where variable of type of array element can be used.

`int a2[5] = {1, 2, 3, 4, 5};`

$\Downarrow$



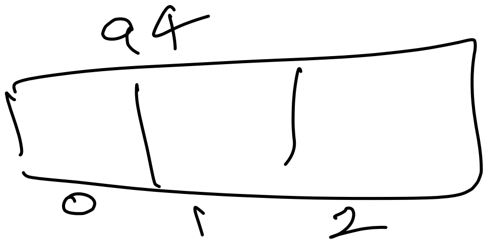
int a3[5] = {1, 2};



int a4[3];

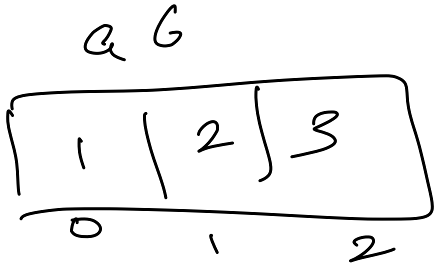
uninitialized array

initial value will be garbage if local,
0 if global or static global or static local.



int a5[ ]; ✗ → syntax error, size missing.

int a6[ ] = {1, 2, 3}; ✓ ⟸ create an array of size = number of element in initializer list.



Q Accept 'n' elements from user, store them in an array.
Find sum of 'n' elements.

```
#include <iostream>

int main( ){

        int nos[100];

        int n;
    do{
        std::cout << "How many nos (<=100)";
        std::cin >> n;
    } while((n >100) || (n <=0));
std:: cout << " Enter " << n << " nos";
for ( int i = 0; i < n; ++i) {
    std:: cin >> nos [i];
}

   int sum = 0;
   for (int i = 0; i < n; ++i) {
        sum = sum + nos[i];    => +=
   }
   std:: cout << "Sum is " << sum;
   return 0;
}
```

sum += nos[i]

Using VLA → Variable Length Array

```
int main() {
    int n;
    std::cin >> n;   // +ve value only

    int nos[n];  ← VLA

     :
     :
}
```

```
#include <iostream>
                    void readElements(int nos[],
                                       int n);
int main() {
    const int MAX_NUMS = 100;
    int nos[MAX_NUMS];

    int n = getValidCount(MAX_NUMS);

    readElements(nos, n);

    int sum = findSum(nos, n);
    std::cout << " Sum is " << endl;
    return 0;

}
```

```
void readElements ( int nos [], int n) {
```

size is not required, as array are not passed as array to function.

```
    for ( int i = 0; i < n; ++i) {
        std :: cin >> nos [i];
    }
}
```