Functions & Arrays

Date

Functions.

↳ block of code → that performs some operation

```
return
 ↓ type
int main()  ⟶ main function
{
   ___
   ___
   ___
   return 0;  ← this shows successful execution.
}
```

function name

argument / parameter

* Explore –
  - → why always 0 is returned at the end of main()?
  - → Agar hum main() ko void return type rakhe, to vo chalega?
    eg void main()

```
{ ___
  ___
}
```

Explore this.

ROCKSTRONG CEMENT

Date ....

Q Why do we need functions?

→ When we need to repeat code/Reuse it.
→ Improves readability.
→ Modular approach.
→ Saves time
→ Clean code
→ Maintainability.
→ Easy to understand
→ Testable
→ Redundancy

If we repeat a piece of code again and again, code gets bulky and it can get buggy too & readability is also reduced.

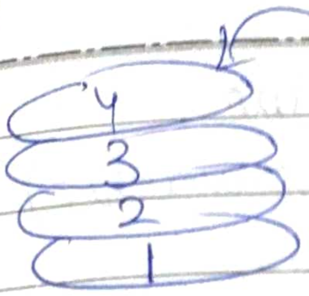
Example -

```
int main()
{
 printNumber()
   }
```

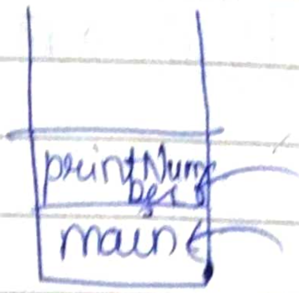
```
void printNumber()
{
cout<<"____";
}
```

Jab hum koi function call krte h, sabse pehle main() ki entry hoti h function call stack me, fir baaki functions ki.

Stack:



insert → 1, 2, 3, 4
extract → 4, 3, 2, 1.



Parameter/Argument of function —

```
main()
{ printNumber(num)
}
```

```
void printNumber(int n)
{
   cout<<n;
}.
```

Value which is passed in the function as input parameters.

* Pass by Value —

```
main()
{
int ~~main~~ num = 5;
   print(num)
   return 0;
}
```

Behind the scene —

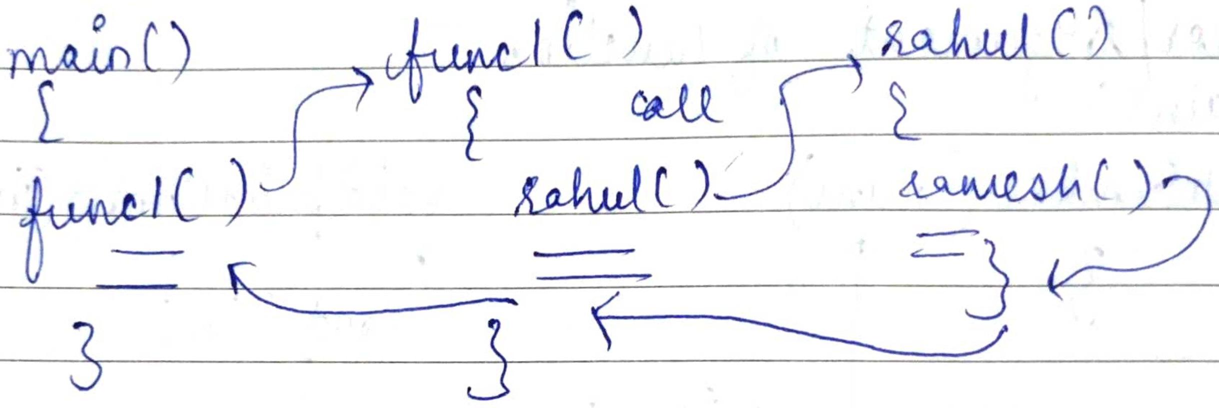
$\xrightarrow{\quad}$ $\boxed{5}$ num — block of memory (in main memory)

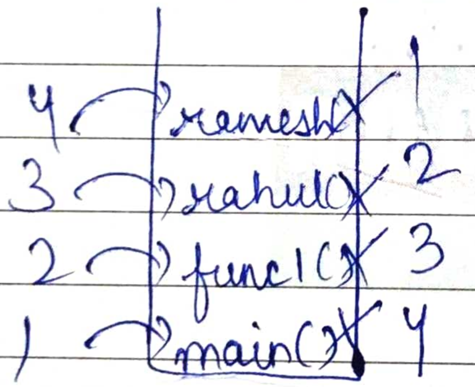
```
print(int n)
{
   cout→ - - -
}
```

$\xrightarrow{\quad}$ $\boxed{5}$ n — another block of memory → copy of num block (in print() memory)

Date

H/W → ~~@~~ 20 MCQs of Pass by Value (GFG, CS or InterviewBit)





Stack :

This is nested call but not recursion.

Explore :→ why (int) main?
→ why no argument in int main()?
→ why return 0 in main() not return 1?

Date

Q I/P n = 15
Print all even no. till n.

```
void
~~int~~ printEven (int n)
{
 for (int i=2; i<=n; i+=2)
   {
     cout << i << " ";
   }
 }

int main ()
{
  int n=15;
  printEven(n);
  return 0;
 }.
```

Q. I/P n = 6
Print all squares till n.

```
void printSquares (int n)
  {
   for (int i=1; i<=n; i++)
        cout << i*i << endl;
  }
```

ROCKSTRONG CEMENT

Date

```
int main()
{
 int n=15;
 printSquares(n);
 return 0;
 }.
```

Q Find factorial of a No. I/P → 5
O/P → 5! = 120

```
void factorial (int n)
 {
 int f = 1;
 for (int i=1; i<=n; i++)
  {
   f = f*i;
  }
  ~~[illegible]~~ cout<<f<<endl;
 }
```

```
int main()
{
 int n=5;
 factorial(n);
 return 0;
  }.
```

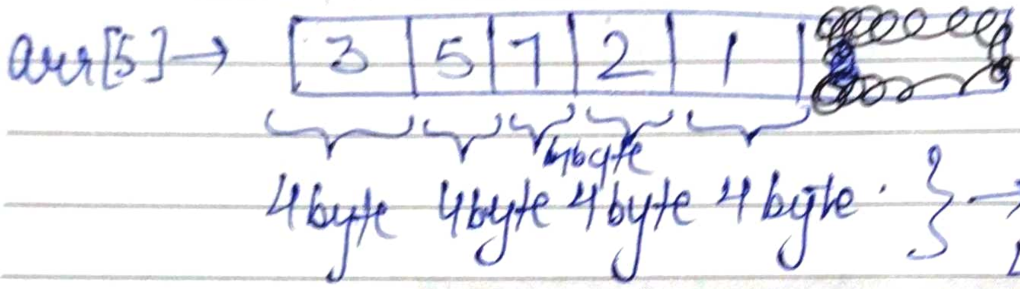
Date

Arrays → Collection of Similar type of elements.

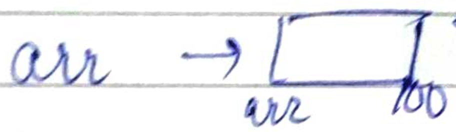
$$\begin{tikzcd}
\text{int arr[5];} \arrow[r, bend left, "\text{name}"] \arrow[r, bend right, "\text{size}"'] & {}
\end{tikzcd}$$

↑ int type ka data

{ Stores data at Continuous memory location }

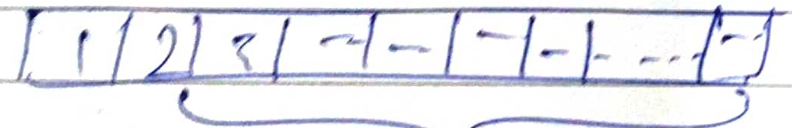


} → total 4×5 20 bytes.



← first block.

int arr[15] = {1, 2}



could be garbage. value or 0 (compiler dependant)

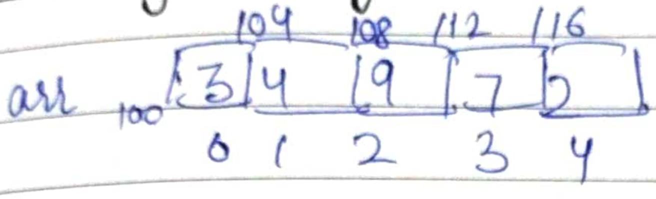
Initializing with single value -

→ int arr[25] = {0};

Date

H/W. check if int arr[30] = {1} will initialize all array indexes with 1.

* Accessing Array Elements -



first element → $\boxed{\text{arr[0]}}$ → 3
arr[3] → 7

# Behind The scene → arr[0] → 100 + 0×4 = $\boxed{100}$

is Location par kya h?

$\boxed{3}$

Formula :-

$$\boxed{\text{arr}[i] = \text{base address} + \text{index} * \text{size of datatype in bytes.}}$$

Q Print elements of array.

```
void printArray (int arr[], int size)
{
for (int i=0; i<size; i++)
  {
    cout << arr[i] << " ";
  }
}
```

```
int main()
 {
  int arr[5] = {3,6,9,2,18};
  printArray(arr,5);
    return 0;
  }
```

Note: Make sure, while passing array in any function, pass the size of array also.

Pass by Reference - When we pass an array as a function parameter, its reference is passed, not copy so there is change in the actual array.

```
void util (int arr[], int size)
  {
  arr[0] = 23;
  cout <<"printing in UTIL" <<endl;
 printArray(arr,3);
  }

int main()
 {
    int arr[] = {3,6,9};
     util(arr,3);
```

Date

```
cout << " Printing in main ";
printArray (arr,3);
}
```

output –
```
23 6 9
23 6 9
```
same in both functions.

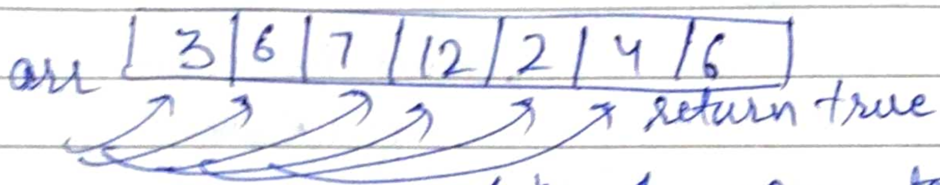
Array number is not copied in a function parameter, but reference pointer is passed. Actual value is changed.
∴ answer is same in both main & util.

Jab kisi array ko function me pass krege or function me value ko update ya change krege to vo change har jagah sustain krega because value pass ke time reference of array jaata h.

4/10 Search about behind the scenes of when we pass an array in a function parameter.

Linear Search :-

target = 4



1 by 1 move towards end of array & send true if no. found

```
bool search(int arr[], int size, int target)
 {
   for (int i=0; i<size; i++)
       {
     if (arr[i]== target)
         return true;
       }
 }

 int main()
  {
    int arr[100];
    cout << "Enter size " <<endl;
     int n;
     cin>> n;
 for (int i=0; i<n; i++)
     {
         cin>> & arr[i];
           }
```

ROCKSTRONG CEMENT

Date

```
if(search(arr,5,7))
 {
   cout<<"element found"<<endl;
   }
 else
   cout<<"not found"<<endl;
 }
return 0;
 }.
```