The concept of operator precedence not seems useful now but it is, it will be used ~~very~~ various times in making complicated programms.

→ Control structure in C++

Control structure give flow, logic & structure to a C++ programme.

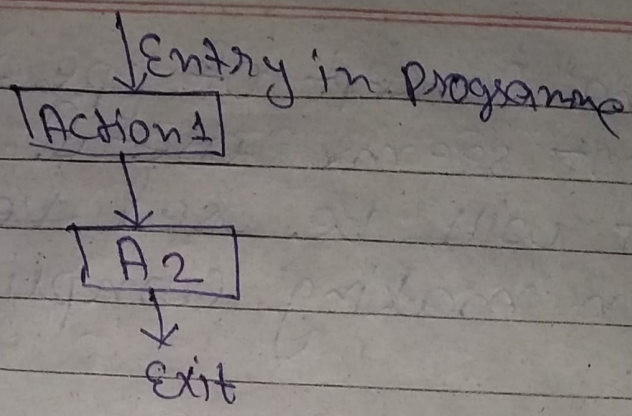
It is 3 types -

→ Sequence structure
→ selection structure
→ loop structure

} Basic control structure

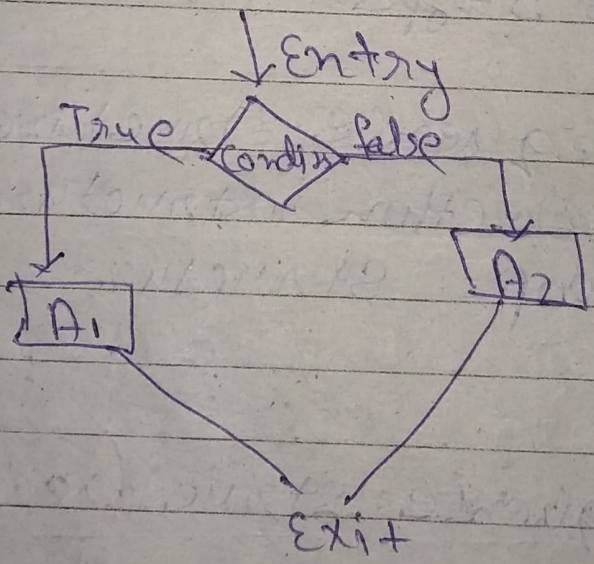
→ Sequence structure :-

It gives a programme sequence, in which ~~programme~~ flow programm operate.

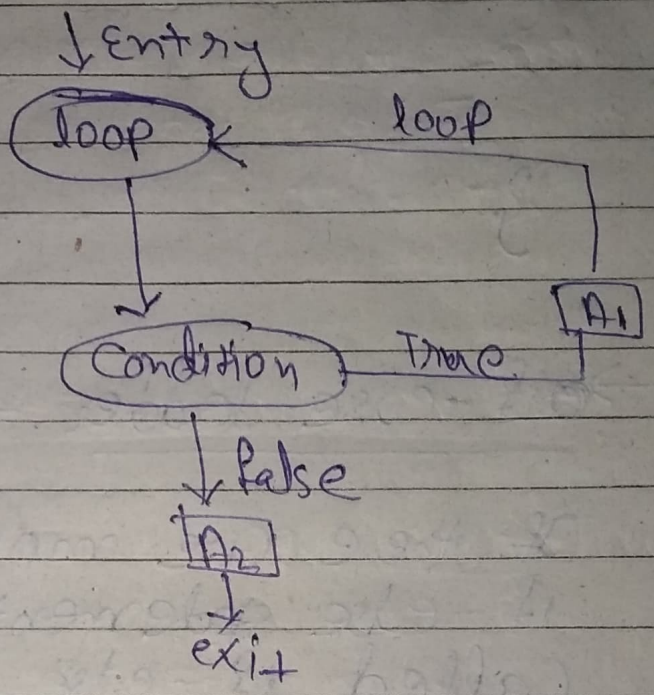


→ Selection structure :-

It used to check condition in programme.



→ loop structure :-



→ if-else statement :-

if-else is a control statement in C++.

Syntax :-

```
if( condition)
{
  =
  =
}
```

```
else
{
  ---
  ==
}
```

→ **if-else ladder :-**

If there are more than one if-else statement then it is called if-else ladder.

```
if (con.)
{
  ==
}
else if (con.)
{
  ==
}
else if (con.)
{
  ==
}
else
{
  ==
}
```

There is no limit how ~~many~~ long if-else ladder can be you can add as many else if() condition as you want b/w if() & else() condition.

→ Switch-case statement:-

```
int x;
Switch (x)
{
  case 3:
  {
    =
    break;
  }
  case 10:
  {
    =
    break;
  }
default:
  {
    =
    break;
```

→ Loops :-

In c++ loops are of 3 types:-

i) for loop
ii) while loop
iii) do-while loop

→ for loop :-

```
for(initialization;condition;increment)
{
    =
    =
    statement
}
```

→ while loop :-

```
while (condition)
{
   statement
}
```

→ do-while loop:-

```
do
{
    statement
}
while(condition);
```

EXTRA<VS>

{

User Snippets:-

It is an option is VS by which you can make basic structure for something like if you write a word which you make snippets it will give ~~open~~ option, as you see when write for and you get syntax like thing.

}

→ break statement :-

break statement is used when you want to break and comeout from a ongoing loop.

ex.-

```
for(int i=0; i<=20; i++)
{
    cout<<i<<endl;
    if(i==2)
    {
        break;
    }
}
```

By writing this code it will only print

0
1
2

→ Continue statement:-

Continue statement is used then it stops the current iteration of loop & starts/continue with next iteration.

ex.-

```
for (int i=0;i<=7;i++)
{
    cout<<i<<endl;
    if(i==2)
    {
        continue;
    }
}
```

output:-

```
0
1
3
4
5
6
7
```

→ Pointers :-

Pointer is a type of variable/ data type which holds/contains address of other data type.

making a pointer variable :-

```
int a;
int * b = & a;
```

where,

& → (address of) operator
* → (de reference) operator.

As of now, we know to store address of a variable we need pointers, but what if we want to store an address of a pointer itself.

For this we use

```
int **c = & b;
```

here double star before 'c' indicate that 'c' stores address of a pointer 'b' itself.

→ Arrays :-

You can define array in two different ways, both has same effect on programme, but both defenation are different in the way to define mechanism and working of array.

First we will see simple difination

Array is a collation of data of same data type stored in contiguous memory location.
for ex.-

1	2	5	7	9

int A[5]

It is an array name A[5] of size 5 and of ~~time~~ type int.

Syntax-

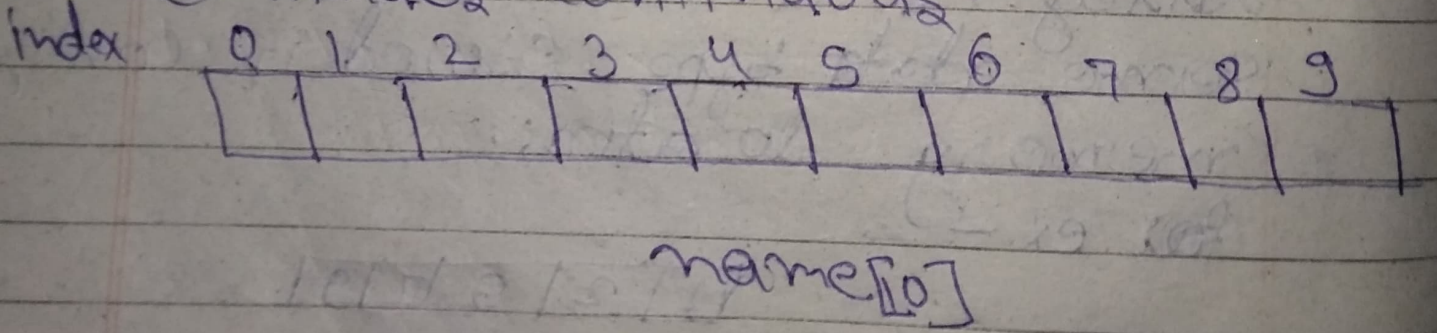
~~defn~~ type name[size];

Second defination is more like it's mechanism and how array works with pointer.

When we define an array. Say

char name[10];

we have made/allocated 10 memory block in memory ~~continues~~ continuous.



name[0]

now, initally we only know address of first index (here 0) of array after that to know

address of next index we add size of data type of array, like here, we will add 1 as size of one char is 1 byte.
1 in the address of previous index.
Let's take another example of array

```
int add[5];
```

now, we have address of first index (index 0) to know address of next index we will add '1' in the address of previous index. ~~size of one int is 2 byte~~

NOTE:- If you defined a pointer variable and assigned it with the name of array then automatically you get address of first index of array. ex-

```
int *p[10];
int *a = p;
```

and after that you can get address of next index as add earlier by adding 1.

→ Structure enums & Unions :-

→ Structure :-

Structure is used to make coustum data type which contains data with different datatype.

We can define structure in two ways.-

1. #include<iostream>
Using namespace std;

Struct employee
{
int id;
char fav;

```
int main()
{
    struct employee shubham;
    struct employee rohan;

    shubham.id = 023;
    shubham.fav = 'c';
    shubham.salary = 23097895;

    cout<<shubham.id<<endl;
    cout<<shubham.fav<<endl;
    cout<<shubham.salary<<endl;

    return 0;
}
```

In this example in place of employee you can choose any word, then that word will be your new data type.

2.

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

typedef struct employee
{
    int id;
    char fav;
    float salary;
} ep;

int main()
{
    ep harry;

    harry.id = 0235;
    harry.fav = 'D';
    harry.salary = 80000000;

    cout<<harry.id<<endl;
    cout<<harry.fav<<endl;
    cout<<harry.salary<<endl;
    return;
}
```

In this example we have defined employee as data type but while writting code it is not practical to write

'struct employee' as data type as it is long, so we use typedef and define a new small word (for here 'ep') as abbriviation so, we can use it as data type.

And it is vary convinient.

→ Unions :-

Unions are same as structure but they provide better memory management than structure. In union, it uses shared memory because of that you can access only one variable at a time.

For example you have a union money and it contains three variable then it allocate only as much memory as, it can store biggest variable at a time.

If you uses two variable of union at one time it automatically deactivate privious one.

ex. -

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

union money
{
    int car;
    char fav;
    float value;
};
int main()
{
```

```
Union money M1;
M1.car= 3;
cout<<M1.car<<endl;
M1.fav = 'c';
cout<<M1.fav<<endl;
return 0;
}
```

In output you can see that value of M1.car is different than what we have given as it is overwritten by M1.fav.