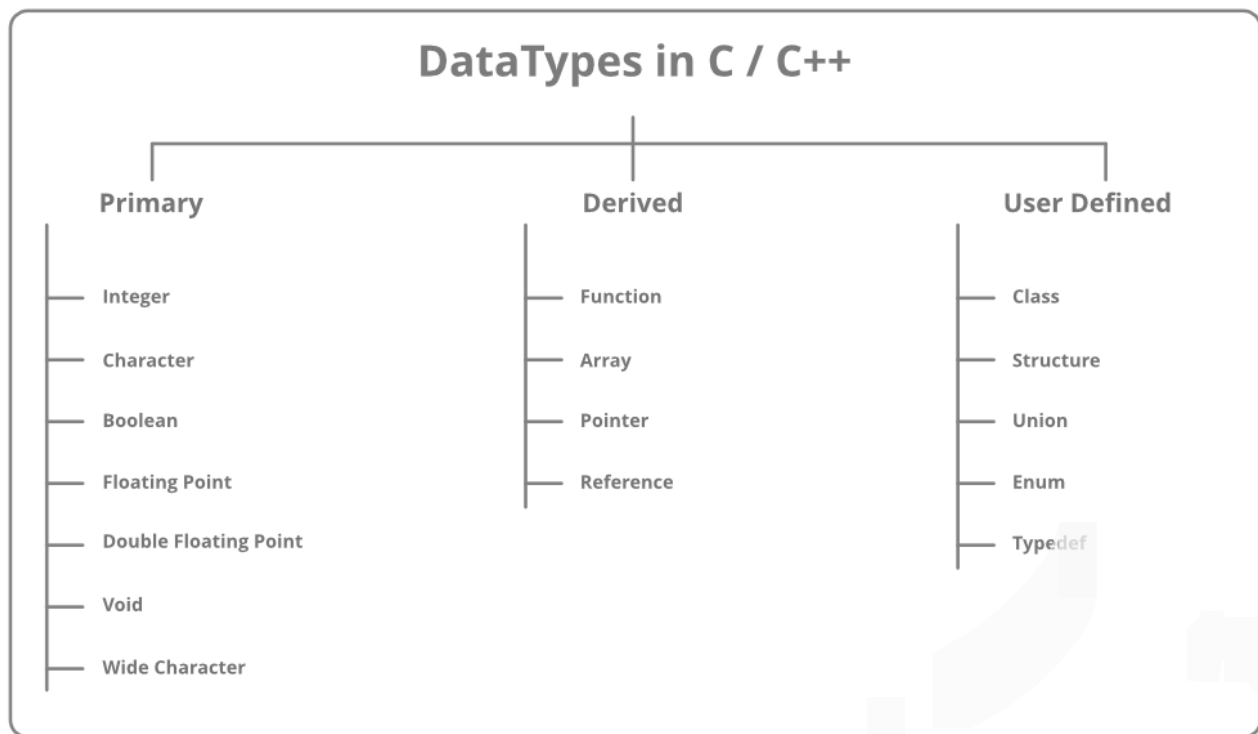


C++ Notes



Data type	Bytes occupied in RAM	Value range	Value range in decimal
char	1 byte	-2^7 to 2^7-1	-128 to 127
int	2 byte	-2^{15} to $2^{15}-1$	-32768 to 32767
float	4 byte	-2^{31} to $2^{31}-1$	3.4e-38 to 3.4e+38
double	8 byte	-2^{63} to $2^{63}-1$	1.7e-308 to 1.7e+308
long double	10 byte	-2^{79} to $2^{79}-1$	3.4e-4932 to 1.1e+4932

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

// main() is where program execution begins.
int main() {
    cout << "Hello World"; // prints Hello World
    return 0;
}
```

Let us look at the various parts of the above program –

- **#include** Preprocessor directive used to include files.
- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header **<iostream>** is needed.
- The line **using namespace std;** tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.

- The next line '**// main() is where program execution begins.**' is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.
- The line **int main()** is the main function where program execution begins.
- The next line **cout << "Hello World";** causes the message "Hello World" to be displayed on the screen.
- The next line **return 0;** terminates **main()** function and causes it to return the value 0 to the calling process.

C++ Keywords

Keywords are the reserved words that have a special meaning to the compiler. They are reserved for a special purpose, which cannot be used as the identifiers. For example, 'for', 'break', 'while', 'if', 'else', etc. are the predefined words where predefined words are those words whose meaning is already known by the compiler.

A list of 32 Keywords in C++ Language which are also available in C language are given below.

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

A list of 30 Keywords in C++ Language which are not available in C language are given below.

asm	dynamic_cast	namespace	reinterpret_cast	bool
explicit	new	static_cast	false	catch
operator	template	friend	private	class
this	inline	public	throw	const_cast
delete	mutable	protected	true	try
typeid	typename	using	virtual	wchar_t

C++ Operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

There are following types of operators to perform different types of operations in C language.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operator
- Unary operator
- Ternary or Conditional Operator
- Misc. Operator

	Operator	Type
Unary operator	+ +, - -	Unary operator
Binary operator	+ , - , * , / , %	Arithmetic operator
	< , <= , > , >= , == , !=	Relational operator
	&& , , !	Logical operator
	& , , << , >> , ~ , ^	Bitwise operator
Ternary operator	= , += , -= , *= , /= , % =	Assignment operator
	?:	Ternary or conditional operator

Precedence of Operators in C++

The precedence of operator species that which operator will be evaluated first and next. The associativity specifies the operators direction to be evaluated, it may be left to right or right to left.

Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Right to left
Shift	<< (Left Shift) >> (Right Shift)	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Right to left
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Right to left
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right

C++ Identifiers

The identifiers are the names which are defined by the programmer to the program elements such as variables, functions, arrays, objects, classes.

The identifier name cannot start with a digit, i.e., the first letter should be alphabetical. After the first letter, we can use letters, digits, or underscores.

Identifiers	Keywords
Identifiers are the names defined by the programmer to the basic elements of a program.	Keywords are the reserved words whose meaning is known by the compiler.
It is used to identify the name of the variable.	It is used to specify the type of entity.
It can consist of letters, digits, and underscore.	It contains only letters.
It can use both lowercase and uppercase letters.	It uses only lowercase letters.
No special character can be used except the underscore.	It cannot contain any special character.
The starting letter of identifiers can be lowercase, uppercase or underscore.	It can be started only with the lowercase letter.
It can be classified as internal and external identifiers.	It cannot be further classified.
Examples are test, result, sum, power, etc.	Examples are 'for', 'if', 'else', 'break', etc.

Bitwise Operator (Left Shift and Right Shift)

Left Shift

$N \ll i$ (N: first operand, i: second operand)

Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift. Or in other words left shifting an integer “x” with an integer “y” denoted as ‘ $x \ll y$ ’ is equivalent to multiplying x with 2^y (2 raised to power y).

e.g. lets take $N=22$; which is 00010110 in Binary Form.

Now, if “N is left-shifted by 2” i.e $N=N \ll 2$ then N will become $N=N*(2^2)$. Thus, $N=22*(2^2)=88$ which can be written as 01011000.

Right Shift :

$N \gg i$ (N: first operand, i: second operand)

Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift. In other words right shifting an integer “x” with an integer “y” denoted as ‘ $x \gg y$ ’ is equivalent to dividing x with 2^y .

eg: lets take $N=32$; which is 100000 in Binary Form.

Now, if “N is right-shifted by 2” i.e $N=N>>2$ then N will become $N=N/(2^2)$. Thus, $N=32/(2^2)=8$ which can be written as 1000.

Important Points :

- The left shift and right shift operators should not be used for negative numbers. The result of is undefined behaviour if any of the operands is a negative number.
For example results of both $-1 << 1$ and $1 << -1$ is undefined.
- If the number is shifted more than the size of integer, the behaviour is undefined. For example, $1 << 33$ is undefined if integers are stored using 32 bits.
For bit shift of larger values $1ULL << 62$ **ULL** is used for Unsigned Long Long which is defined using 64 bits which can store large values.
- The left-shift by 1 and right-shift by 1 are equivalent to the product of first term and 2 to the power given element($1 << 3 = 1 * \text{pow}(2,3)$) and division of first term and second term raised to power 2 ($1 >> 3 = 1 / \text{pow}(2,3)$) respectively.

C++ Basic Input/Output

I/O Library Header Files

Let us see the common header files used in C++ programming are:

Header File	Function and Description
<iostream>	It is used to define the cout , cin and cerr objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.
<iomanip>	It is used to declare services useful for performing formatted I/O, such as setprecision and setw .
<fstream>	It is used to declare services for user-controlled file processing.

Standard output stream (cout)

The **cout** is a predefined object of **ostream** class. It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream **insertion operator** (<<) to display the output on a console

Standard input stream (cin)

The **cin** is a predefined object of **istream** class. It is connected with the standard input device, which is usually a keyboard. The cin is used in conjunction with stream **extraction operator** (>>) to read the input from a console.

Standard end line (endl)

The **endl** is a predefined object of **ostream** class. It is used to insert a new line character and flushes the stream.

“\n” only insert a new line.

“endl” doesn’t occupy any memory whereas “\n” is a character, so it occupies 1 byte memory

“endl” is used in C++ only

“endl” = \n + flush()

```
#include <iostream>
using namespace std;
int main( ) {
    int age;
    cout << "Enter your age: ";
    cin >> age;
    cout << "Your age is: " << age << endl;
}
```

Extraction Operator

Insertion Operator

Some Other Header Files

stdio.h/cstdio: It is the Standard C input and Output, as printf, scanf, fprintf etc. This API is a ‘c’ API only and in C++ avoided

iostream: Standard C++ input and output contains object like cout, cin and cerr. Ans in C++ iostream are still default choice for most of the project.

conio.h: Stands for “console input/output”, a window only header which provides C function for console IO manipulation, like getch, ungetch etc.

C++ Control Statement

In C++ programming, **if statement** is used to test the condition. There are various types of if statements in C++.

- **if statement**

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

int main () {
    int num = 10;
    if (num % 2 == 0)
    {
        cout<<"It is even number";
    }
    return 0;
}
```

- **if-else statement**

```
#include <iostream>
using namespace std;
int main () {
    int num;
    cout<<"Enter a Number: ";
    cin>>num;
    if (num % 2 == 0)
    {
        cout<<"It is even number"<<endl;
    }
    else
    {
        cout<<"It is odd number"<<endl;
    }
    return 0;
}
```

- **nested if statement**

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

int main()
{
    int i = 10;
    if (i == 10)
    {
        if (i < 15)
            cout<<"i is smaller than 15\n";
        if (i < 12)
            cout<<"i is smaller than 12 too\n";
        else
            cout<<"i is greater than 15";
    }
    return 0;
}
```

- **if-else-if ladder**

```
#include <iostream>
using namespace std;
int main () {
    int num;
    cout<<"Enter a number to check grade:";
    cin>>num;
    if (num <0 || num >100)
    {
        cout<<"wrong number";
    }
    else if(num >= 0 && num < 50){
        cout<<"Fail";
    }
    else if (num >= 50 && num < 60)
    {
        cout<<"D Grade";
    }
    else if (num >= 60 && num < 70)
    {
        cout<<"C Grade";
    }
    else if (num >= 70 && num < 80)
    {
        cout<<"B Grade";
    }
    else if (num >= 80 && num < 90)
    {
        cout<<"A Grade";
    }
    else if (num >= 90 && num <= 100)
    {
        cout<<"A+ Grade";
    }
}
```

The C++ **switch statement** executes one statement from multiple conditions. It is like if-else-if ladder statement in C++.

```
#include <iostream>
using namespace std;
int main () {
    int num;
    cout<<"Enter a number to check grade:";
    cin>>num;
    switch (num)
    {
        case 10: cout<<"It is 10"; break;
        case 20: cout<<"It is 20"; break;
        case 30: cout<<"It is 30"; break;
        default: cout<<"Not 10, 20 or 30"; break;
    }
}
```

Output

Enter a number:

10

It is 10

→ If none of the condition is match,
Then, this line will print...

The C++ for loop is used to iterate a part of the program several times. If the number of iterations is fixed, it is recommended to use for loop than while or do-while loops.

For Loop

```
#include <iostream>
using namespace std;
int main() {
    for(int i=1;i<=10;i++){
        cout<<i <<"\n";
    }
}
```

Output

1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3

Nested For Loop

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

int main () {
    for(int i=1;i<=3;i++){
        for(int j=1;j<=3;j++){
            cout<<i<<" "<<j<<"\n";
        }
    }
}
```

Infinite For Loop

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

int main () {
    for (; ; )
    {
        cout<<"Infinitive For Loop";
    }
}
```

Output

Infinitive For Loop
Infinitive For Loop
Infinitive For Loop
Infinitive For Loop
Infinitive For Loop
ctrl+c

In C++, **while loop** is used to iterate a part of the program several times. If the number of iterations is not fixed, it is recommended to use while loop than for loop.

While Loop

```
#include <iostream>
using namespace std;
int main() {
    int i=1;
    while(i<=10)
    {
        cout<<i <<"\n";
        i++;
    }
}
```

Nested While Loop

```
include <iostream>
using namespace std;
int main () {
    int i=1;
    while(i<=3)
```

```

    {
        int j = 1;
        while (j <= 3)
        {
            cout<<i<<" "<<j<<"\n";
            j++;
        }
        i++;
    }
}

```

Infinite While Loop

```

#include <iostream>
using namespace std;
int main () {
    while(true)
    {
        cout<<"Infinitive While Loop";
    }
}

```

The C++ **do-while** loop is used to iterate a part of the program several times. If the number of iterations is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The C++ do-while loop is executed at least once because condition is checked after loop body.

do-while Loop

```

#include <iostream>
using namespace std;
int main() {
    int i = 1;
    do{
        cout<<i<<"\n";
        i++;
    } while (i <= 10);
}

```

Nested do-while Loop

```

#include <iostream>
using namespace std;
int main() {
    int i = 1;
    do{
        int j = 1;
        do{
            cout<<i<<"\n";
            j++;
        } while (j <= 3);
        i++;
    } while (i <= 3);
}

```

Infinitive do-while Loop

```
#include <iostream>
using namespace std;
int main() {
    do{
        cout<<"Infinitive do-while Loop";
    } while(true);
}
```

The C++ **break** is used to break loop or switch statement. It breaks the current flow of the program at the given condition. In case of inner loop, it breaks only inner loop.

Let's see a simple example of C++ break statement which is used **inside the loop**.

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 10; i++)
    {
        if (i == 5)
        {
            break;
        }
        cout<<i<<"\n";
    }
}
```

The C++ break statement breaks inner loop only if you use break statement **inside the inner loop**.

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<=3;i++){
        for(int j=1;j<=3;j++){
            if(i==2&& j==2){
                break;
            }
            cout<<i<<" "<<j<<"\n";
        }
    }
}
```

Output

1 1
1 2
1 3
2 1
3 1
3 2
3 3

The C++ **continue statement** is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

```
#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<=10;i++){
        if(i==5){
```

```

        continue;
    }
    cout<<i<<"\n";
}
}


```

C++ Continue Statement continues inner loop only if you use continue statement **inside the inner loop**.

```

#include <iostream>
using namespace std;
int main()
{
    for(int i=1;i<=3;i++){
        for(int j=1;j<=3;j++){
            if(i==2&&j==2){
                continue;
            }
            cout<<i<<" "<<j<<"\n";
        }
    }
}

```



Output
1 1
1 2
1 3
2 1
2 3
3 1
3 2
3 3