Basics of C++

- C++ was developed in “Bell Lab” during 1980s by “Bjarne Straustrup”.
- C++ is a superset of C language. It includes some new non-object-oriented features to enhance the capability of C and object- oriented features to make programming object based.
- It includes some new keywords which have been listed below-

asm	auto	bool	break
case	catch	char	class
const	const_cast	continue	default
delete	do	double	dynamic_cast
else	enum	explicit	export
extern	false	float	for
friend	goto	if	inline
int	long	mutable	namespace
new	operator	private	protected
public	register	reinterpret_cast	return
short	signed	sizeof	static
static_cast	struct	switch	template
this	throw	true	try
typedef	typeid	typename	union
unsigned	using	virtual	void
volatile	wchar_t	while	

Components of C++ programs:

```
#include<iostream.
using namespace std;
int main()
{
    int a , b, c;
    cout<<"Enter the values of a and b"<<endl;
    cin>>a>>b;
    c=a+b;
    cout<<"The sum is "<<c<<endl;
    return 0
}
```

- *cout* is an object (**standard output stream**).
- << is an operator called as **Insertion** or **put to**. It puts the content of the variable on its right-hand side to the *cout* object
- Even though << is called as the **Bitwise left-shift operator**, due to the concept of function overloading, its interpretation is different here.

- *cin* is also an object (**standard input stream**) that represents the stream coming from the keyboard.
- >> is called as the **extraction** or **get from** operator. It extracts the content from the *cin* object on its left-hand side and put it in the variables in its right
- '**iostream.h**' contains the declarations that are needed by *cout*, *cin*, << and >>.
- **Name space** is a collection of identifiers that all belong to a group or family. All identifiers in the C++ standard library belongs to a name space std.
- If we do not use the statement "*using namespace std ;*", then we need to include *std* with all the identifiers belong to this namespace.

  For example-

  std::cout<<"Enter the values of a and b"<<endl;

- "***endl***" is known as manipulator in C++. It causes a linefeed to be inserted in the output stream. As a result, the phrase following it appears on a fresh line. (its purpose is similar to that of "***n***" in C.

Function Prototypes:

- Function prototype is a declaration that specifies the number, order and types of arguments for a function.

Example-

```
int add(int , int );
float square(float);
char * strconvert(char*, int)
```

- No C++ function can be called unless its prototypes is available to the compiler to crosscheck the argument type and the return value. This is called string type checking.

Example 1.

```
/* C++ Function Prototype and C++ Function Definition */

#include<iostream.h>
#include<conio.h>
#include<stdlib.h>

int add(int, int);          // function prototype
int subtract(int, int);     // function prototype
int multiply(int, int);     // function prototype
int divide(int, int);       // function prototype

void main()
{
```

```
   clrscr();
   int a, b;
   cout<<"Enter any two number: ";
   cin>>a>>b;
   cout<<"\nSummation = "<<add(a, b);
   cout<<"\nSubtraction = "<<subtract(a, b);
   cout<<"\nMultiplication = "<<multiply(a, b);
   cout<<"\nDivision = "<<divide(a, b);
   getch();
}

int add(int x, int y)     // function definition
{
   int res;
   res = x + y;
   return res;
}

int subtract(int x, int y)    // function definition
{
   int res;
   res = x - y;
   return res;
}

int multiply(int x, int y)    // function definition
{
   int res;
   res = x * y;
   return res;
}

int divide(int x, int y)    // function definition
{
   if(y==0)
   {
      cout<<"\n\nDivide by Zero Error..!!";
      cout<<"\nPress any key to exit...";
      getch();
      exit(1);
   }
   else
   {
      int res;
      res = x / y;
      return res;
   }
}
```

Example 2.

```
#include<string>  // Contains the prototype   for strupr() function
#include<iostream>
using namespace std;
int main()
{
    char str[] = "Gaya College of Engineering";
    strupr(str);
    cout<<str<<endl;
}
```

Flexibility in declaration:

C++ allows definition (declaration) of variables at the point where they are used.

Example:

```
#include<iostream.
using namespace std;
int main()
{
    int f;
    cin>>f;
    int c = (f-32)*5/9;
    cout<<c;
    for(int i = 0;i<6;i++)
    {
        cout<<endl<<i<<endl;

    }
    return 0;
}
```

structure, union and enum:

structure is an user-defined data types (We will learn more on data types in coming classes) in C++. Structure is used to group or store dissimilar data types.

Example:

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

struct Person
{
    char name[50];
    int age;
    float salary;
};

int main()
{
    Person p1;

    cout << "Enter Full name: ";
    cin.get(p1.name, 50);
    cout << "Enter age: ";
    cin >> p1.age;
    cout << "Enter salary: ";
    cin >> p1.salary;
```

```
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p1.name << endl;
    cout <<"Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary;

    return 0;
}
```

Union is also used to store dissimilar datatypes. In union all the variables share the same memory location. Apart from this the union in C++ also supports functions inside.

Example:

```
#include<iostream>
using namespace std;
union book{
  int sr;
  char name[20];
  float price;
 };

int main()
{
   book b;

  cout<<"Enter Serial No.: ";
  cin>>b.sr;
  cout<<"Enter Book Name : ";
  gets(b.name);
  cout<<"Enter Book Price: ";
  cin>>b.price;

  cout<<"\nSerial No.: "<<b.sr;    // garbage value will be print
  cout<<"\nBook Name : "<<b.name;   // garbage value will be print
  cout<<"\nBook Price: "<<b.price;  // accurate value be print
return 0;
}
```

Enumeration (enum) in C++ is a user defined data types that contains named values (elements, members). These elements represent some integral constant.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // Defining enum Gender
```

```
    enum Gender { Male, Female };

    // Creating Gender type variable
    Gender gender = Male;

    switch (gender)
    {
    case Male:
        cout << "Gender is Male";
        break;
    case Female:
        cout << "Gender is Female";
        break;
    default:
        cout << "Value can be Male or Female";
    }
    return 0;
}
```

Anonymous union and enum:

- An anonymous union does not have a union name (tag), and its elements can be accessed directly without using a *union* variable.
- For a *union* to qualify as an anonymous union, the declaration must not declare a variable of *union* type.

```
// Demonstrate an anonymous union.
#include <iostream>
using namespace std;
int main()
{

  // this is an anonymous union
  union {
    short int count;
    char ch[2];
  };

  // Here, refer to union members directly
  ch[0] = 'X';
  ch[1] = 'Y';
  cout << "union as chars: " << ch[0] << ch[1] << '\n';
  cout << "union as integer: " << count << '\n';

  return 0;
}
```

Anonymous enum:

```
Enum {first, second, slleper, actwotier}
Int t = second
```