INTRODUCTION

C++

Objective

- History of C++
- Facts
- Features of C++

History of C++

- Bjarne Stroustrup
- 1979
- AT & T's Bell Labs
- C with classes
- 1983

Facts

- C++'s OOP aspect was inspired by a computer simulation language called Simula67
- JAVA is written in C++
- Major operating systems of modern times are written in C++
- C++ is worlds 4th most used programming language

Features of C++

- C++ is a middle level language
- C++ supports principles of object oriented paradigm
- C++ joins three separate programming traditions
  - the **procedural language** tradition, represented by C;
  - the **object-oriented** language tradition, represented by the class enhancements C++ adds to C;
  - **generic programming**, supported by C++ templates

Comparison between C and C++

- C++ is a super set of C language
- C++ programs can use existing C software libraries
- C follows top down approach of programming
- C++ follows bottom up approach of programming
- C adopts Procedure Oriented Programming
- C++ adopts Object Oriented Programming

Object Oriented Programming

- OOPs is a programming approach which revolves around the concept of “Object”.

- Any entity in the system that can be defined as a set of properties and set of operations performed using entity's property set, is known as Object.

Object Oriented Programming

- Encapsulation
- Data Hiding
- Abstraction
- Polymorphism
- Inheritance

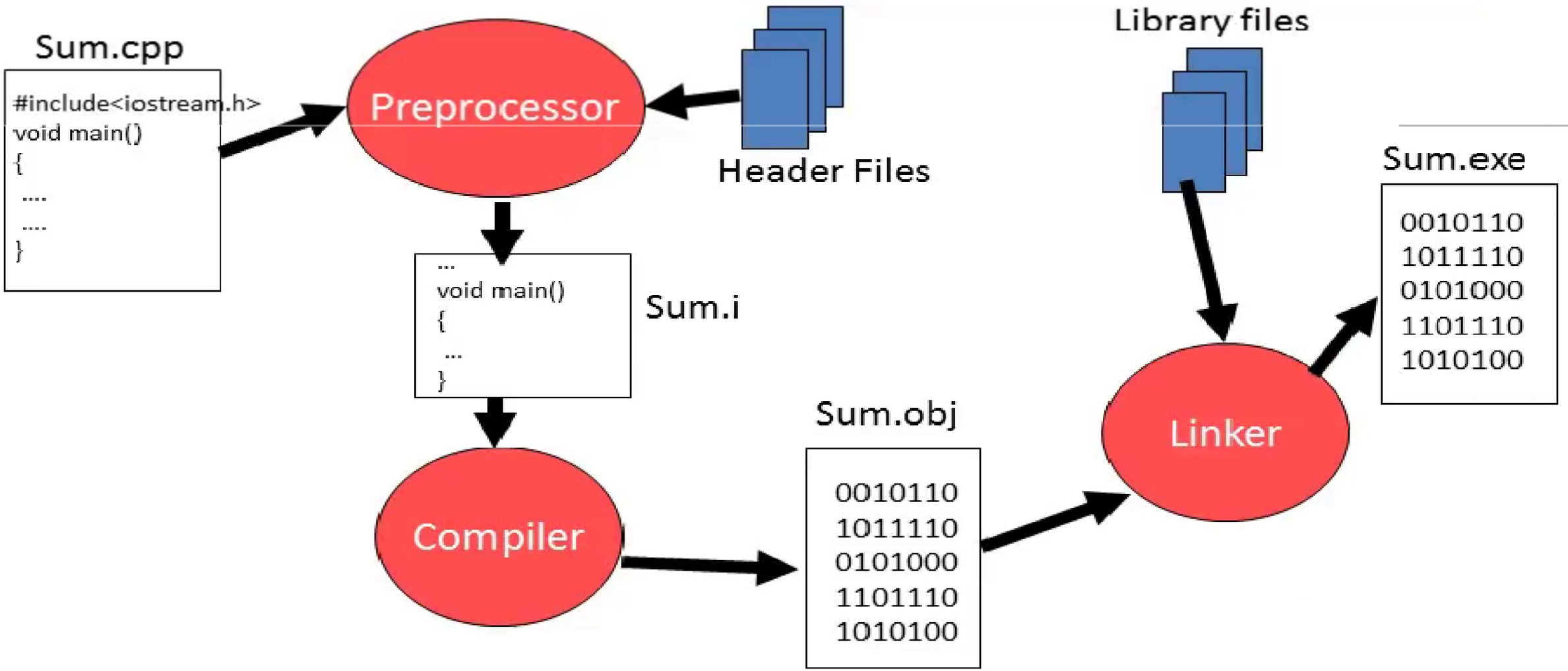
Objective

- Classes and Objects

Concept of Classes and Objects

- Class is a blueprint of an Object
- Class is a description of Object's property set and set of operations
- Creating class is as good as defining a new data type
- Class is a means to achieve encapsulation
- Object is a run time entity
- Object is an instance of a class

Software Development in C++



C++ Constants, Variables & Keywords

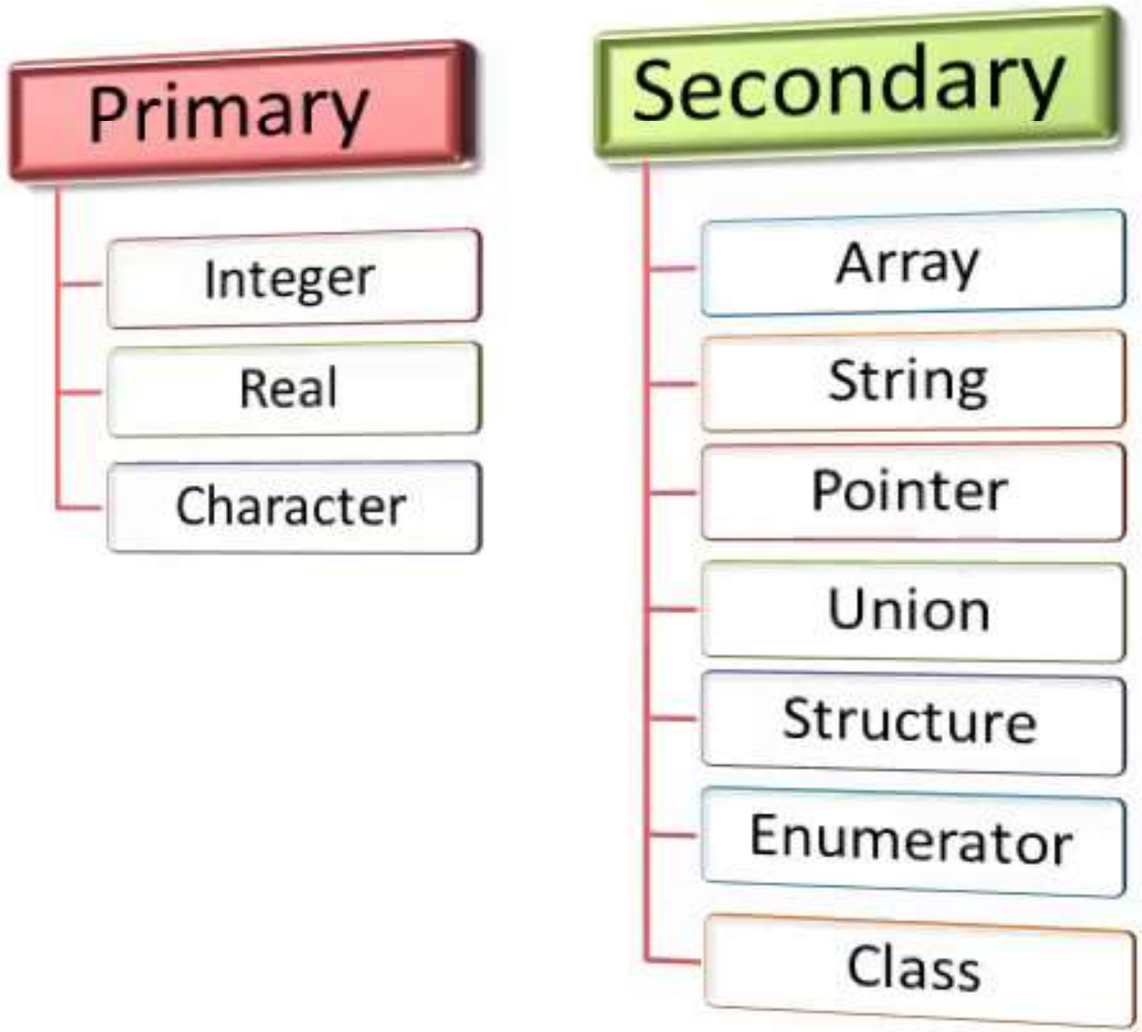
Objective

- Constants
- Variables
- Keywords

Constants

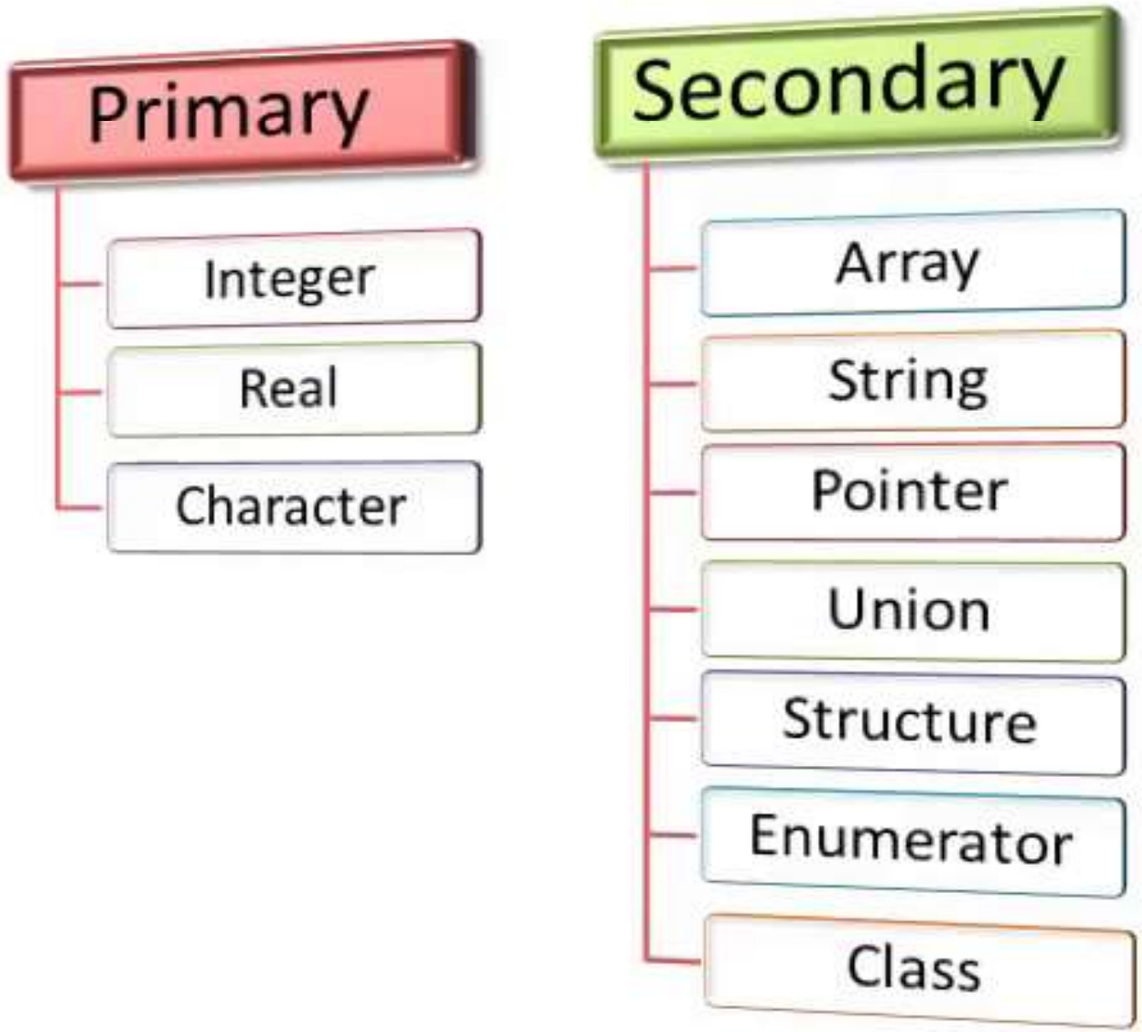
- Any information is constant
- Data = Information = constant

Types of Constants



Types of Constants

- 23 , -341, 0, 5
- 3.4, -0.06, 3.0
- ‘a’, ‘A’, ‘+’, ‘2‘, ‘ ‘



Variables

- Variables are the names of memory locations where we store data.
- Variable name is any combination of alphabet (a to z or A to Z), digit (0 to 9) and underscore (_)

Variables

- Variables are the names of memory locations where we store data.
- Variable name is any combination of alphabet (a to z or A to Z), digit (0 to 9) and underscore (_)
- Valid variable name cannot start with digit

Keywords

auto	double	int	struct	asm	private
break	else	long	switch	catch	public
case	enum	register	typedef	class	protected
char	extern	return	union	delete	template
const	float	short	unsigned	friend	this
continue	for	signed	void	inline	throw
default	goto	sizeof	volatile	new	try
do	if	static	while	operator	virtual

C++ DATA TYPES

Data Type

- int integer
- char character
- float real
- double real

Data types

- int
- char
- float
- double
- void

Data types

- int
- char
- float
- double
- void

```
int a, b=5;

char ch='a';

float k=3.45;

double d1;
```

C++ Fundamental Data Types

The table below shows the fundamental data types, their meaning, and their sizes (in bytes):

Data Type	Meaning	Size (in Bytes)
`int`	Integer	2 or 4
`float`	Floating-point	4
`double`	Double Floating-point	8
`char`	Character	1
`wchar_t`	Wide Character	2
`bool`	Boolean	1
`void`	Empty	0

C++ Modified Data Types List

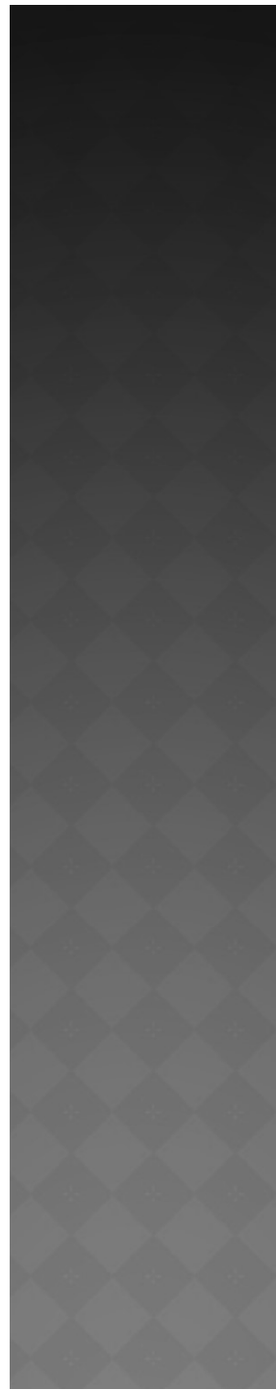
Data Type	Size (in Bytes)	Meaning
`signed int`	4	used for integers (equivalent to `int`)
`unsigned int`	4	can only store positive integers
`short`	2	used for small integers (range **-32768 to 32767**)
`long`	at least 4	used for large integers (equivalent to `long int`)
`unsigned long`	4	used for large positive integers or 0 (equivalent to `unsigned long int`)
`long long`	8	used for very large integers (equivalent to `long long int`).
`unsigned long long`	8	used for very large positive integers or 0 (equivalent to `unsigned long long int`)

Declare Variables Anywhere

- Unlike C, you can declare variables even after action statements.
- Example

```
{
 clrscr();
  int x=4;
}
```

Escape Sequences	Characters
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\‘	Single quotation mark
\“	Double quotation mark
\?	Question mark
\0	Null Character



Sample Program (square.cpp)

```
#include<iostream.h>
#include<conio.h>
void main()
{
   clrscr();
   int x;
   cout<<"Enter a number"<<endl;
   cin>>x;
   int s=x*x;
   cout<<"Square of "<<x<<" is "<<s;
   getch();
}
```

Declaration of cin, cout

Declaration of clrscr(), getch()

Declaration after action statement

Dynamic initialization

Action statement

C++ Basic Input/Output

Output Instruction

- In C, standard output device is monitor and **printf()** is use to send data/message to monitor.
- **printf()** is a predefined function
- In C++, we can use **cout** to send data/message to monitor.

Output Instruction

- cout is a predefined object

- The operator << is called the insertion or put to operator

Output Instruction

```
printf("Hello SCA");
cout<<"Hello SCA";

printf("sum of %d and %d is %d", a, b, c);
cout<<"sum of "<<a<<" and "<<b<<" is "<<c;

printf("%d",a+b);
cout<<a+b;
```

Input Instruction

- In C, standard input device is keyboard and **scanf()** is use to receive data from keyboard
- **scanf()** is a predefined function
- In C++, we can use **cin** to input data from keyboard

Input Instruction

- The identifier cin is a predefined object in C++
- The operator >> is known as **extraction** or get from operator

Input Instruction

```
scanf("%d", &a);
cin>>a;

scanf("%d%d", &a, &b);
cin>>a>>b;

scanf("%d%f", &a, &c);
cin>>a>>c;
```

Header Files

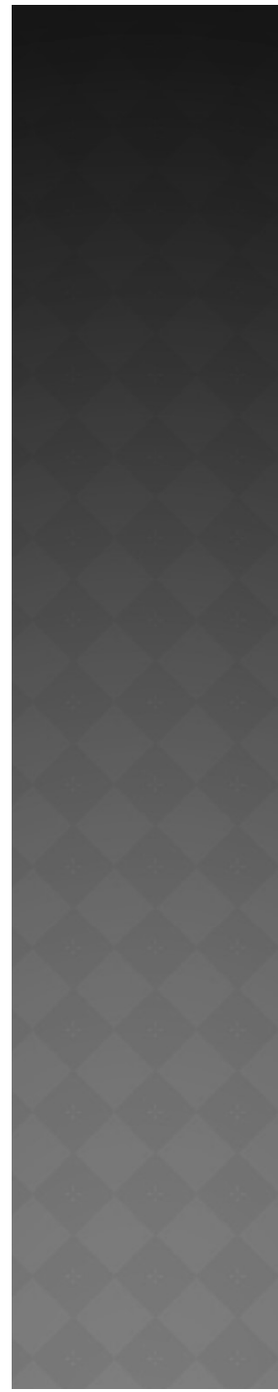
- Predefined functions are declared in header files, so whenever you are using any predefined function in your code, you have to include specific header file that contains its declaration.

STDIO.H

Sr.No.	Functions & Description
1	printf() It is used to print the strings, integer, character etc on the output screen.
2	scanf() It reads the character, string, integer etc from the keyboard.
3	getc() It reads the character from the file.
4	putc() It writes the character to the file.
5	fopen() It opens the file and all file handling functions are defined in stdio.h header file.
6	fclose() It closes the opened file.
7	remove() It deletes the file.
8	fflush() It flushes the file.

IOSTREAM.H

Sr.No.	Functions & Description
1	printf() It is used to print the strings, integer, character etc on the output screen.
2	scanf() It reads the character, string, integer etc from the keyboard.
3	getc() It reads the character from the file.
4	putc() It writes the character to the file.
5	fopen() It opens the file and all file handling functions are defined in stdio.h header file.
6	fclose() It closes the opened file.
7	remove() It deletes the file.
8	fflush() It flushes the file.



About iostream.h

- We need to include header file iostream.h, it contains declarations for the identifier cout and the operator <<. And also for the identifier cin and operator >>.
- Header file contains declaration of identifiers
- Identifiers can be function names, variables, objects, Macros etc

endl

- Inserting endl into the output stream causes the screen cursor to move to the beginning of the next line.
- endl is a manipulator and it is declared in iostream.h
- '\n' character also works as it works in C

Simple program to print string

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
cout << “Viva Technologies";
getch();
}
```

```
int main()
{
    int a,b,c;
    cout<<"enter first number"<<endl;
cin>>a;
cout<<"enter second number"<<endl;
cin>>b;
c=a+b;
cout<<c<<endl;
    return 0;
}
```

C PROGRAM- ADDITION

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
cout<<“Enter Two No. for Addition: “<<endl;
cin>>a>>b;
c=a+b;
cout<<“Answer = “<<c<<endl;
getch();
}
```

TYPE CONVERSION

C++ TYPE CONVERSION

- C++ allows us to convert data of one type to that of another. This is known as type conversion.

There are two types of type conversion in C++.

- Implicit Conversion
- Explicit Conversion

  (also known as Type Casting)

Implicit Type Conversion

- The type conversion that is done automatically done by the compiler is known as implicit type conversion.
- This type of conversion is also known as automatic conversion.

Conversion From int to double

```
#include <iostream>
main()
{
int a = 9;
double b;
b = a;
cout << “a = " << a << endl;
cout << “b = " << b << endl;
}
```

Output

a = 9

b = 9

Automatic Conversion from double to int

```
main()
{
int a;
double b = 9.99;
a = b;
cout << "a = " << a << endl;
cout << "b = " << b << endl;
}
```

Output

a = 9

b = 9.99

C++ Explicit Conversion

- When the user manually changes data from one type to another, this is known as **explicit conversion**.
- This type of conversion is also known as **type casting**.
- There are three major ways in which we can use explicit conversion in C++. They are:
- C-style type casting (also known as **cast notation**)
- Function notation (also known as **old C++ style type casting**)

Type conversion operators

C-style Type Casting

```
int a = 26;
double b;
b = (double)a;
```

Function-style Casting

```
int a = 26;
double b;
b = double(a);
```

Type Casting

```
main()
{
double a = 3.56;
cout << "a = " << a << endl;
// C-style conversion from double to int
int b = (int)a;
cout << "b = " << b << endl;
// function-style conversion from double to int
int c = int(a);
cout << "c = " << c << endl;
}
```

Output

```
a = 3.56
b = 3
c = 3
```

THANKS