Automatic Type Conversion and Type Casting: C++ allows automatic (implicit) type conversion as well as explicit type conversion (type casting) for changing the data type from one to another.

Automatic Type Conversion: C++ has implicit way of converting one datatype to another. The same is done by any of the two ways:

- By assigning a variable of one datatype to another variable of another datatype
- By performing arithmetic operation on two operands of different datatypes, in such case data type of result is automatically in the higher datatype out of the two.

Implicit conversion table

Operand1	Operand2	Result
char	char	char
char	int	int
int	int	int
int	float	float
float	float	float
float	double	double
double	double	double

Automatic Type Conversion Examples

```
char CH='A';
int Number=CH;
cout<<Number<<endl;
cout<<CH<<endl;
Number+=CH;
cout<<Number<<endl;
```

Output
65
A
130

```
int A=1,B=2;
float C=A/B;
cout<<C<<endl;
cout<<A/B<<endl;
cout<<1.0/B<<endl;
cout<<A/2.0<<endl;
```

Output
0
0
0.5
0.5

Type Casting: It is a way to convert one data type to another explicitly by mentioning the data type in braces in front of the operand (Variable/Expression).

Type Casting Examples

```
char CH='A';
int Number;
Number=CH;
cout<<(char)Number<<endl;
cout<<(int)CH<<endl;
```

Output
A
65

```
int A=1,B=2;
float C;
C=(float)A/B;
cout<<C<<endl;
C=A/(float)B;
cout<<C<<endl;
```

Output
0.5
0.5

Formatting of output: In C++, we can format the output with the help of setw() in cout. setw() is used to set width of the next value. It is present in **iomanip** header file.

Example:

```
int Rno1=56,Rno2=5,Rno3=324;
float Fees1=5600,Fees2=940,Fees3=50;
cout<<setw(5)<<Rno1<<setw(8)<<"Anu"<<setw(5)<<Fees1<<endl;
cout<<setw(5)<<Rno2<<setw(8)<<"Tarini"<<setw(5)<<Fees2<<endl;
cout<<setw(5)<<Rno3<<setw(8)<<"Jatin"<<setw(5)<<Fees3<<endl;
```

Output:

```
   56     Anu 5600
    5  Tarini  940
  324   Jatin   50
```

Note: The values are right aligned in the width reserved for the value. setw() does not play any role, if the number digits in the value are higher than width reserved for it.

Comment Line: In C++, we can write some non-executable lines for describing the program or program statements. These lines are known as comment lines. C++ has two ways to add comments into source code. (1) Single Line Comment and (2) Multi-Line Comment

Any line or portion of line that starts with a // is a Single Line Comment. Whereas a Multi-Line Comment starts with /* and ends with */

Example:

```
/*
This program is used for Fahrenheit to Celsius
Developed By: John Amit Mahmood
Date        : 14-August-2013
*/
#include <iostream.h>
void main()
{
  float F,C;
  cout<<"Temperature in Fahrenheit:";cin>>F;
  C=(F-32)/1.8;  //Formula for converting Fahrenheit to Celsius
  cout<<"Temperature in Celsius:"<<C<<endl;
}
```

Instant character input: In C++, you already have seen entering values with the help of **cin>>**. While using **cin>>**, you must have noticed that you need to press <Enter> key after entering the value. If you wish to enter a character value and want it to be instantly accepted and assigned to a char variable without pressing <Enter> key, you can use **cin.get()**. It will also allow you to even accept space as a character value, which is not acceptable while using **cin>>**

Example:

```
char CH;
cout<<"Press Any Key on the Keyboard:";
CH=cin.get();
cout<<"ASCII Value of the Key Pressed:"<<(int)CH<<endl;
```

Control Structure: Branching and Looping are two control structures used in programming languages. Branching and looping, both require conditions. In C++, the following relational operators and logical operators are used in conditions.

Relational Operators: Used to compare values of two operands.

Operator	Description	Sample Condition	Description of Sample Condition
==	Equal To	A==B	Will be TRUE if values of A and B are equal
>	Greater Than	A>B	Will be TRUE if value of A is greater than B
<	Less Than	A<B	Will be TRUE if value of A is less than B
>=	Greater or Equal To	A>=B	Will be TRUE if value of A is greater than or equal to B
<=	Less or Equal To	A<=B	Will be TRUE if value of A is less than or equal to B
!=	Not Equal To	A!=B	Will be TRUE if value of A is not equal to B

Logical Operators: Used to combine two conditions or negate a condition. While execution, it follows order of precedence as (),!,&&,|| (i.e.Brackets, NOT, AND, OR)

Operator	Description	Sample Compound Condition	Description of Sample Compound Condition
&&	AND	Age>=60 && Gender=='M'	Will be TRUE if both conditions are TRUE
||	OR	Age>=60 || Gender=='M'	Will be TRUE if any one of/both the condition(s) is/are TRUE
!	NOT	!(Amount>1000)	Will be TRUE if condition mentioned in the brackets is NOT TRUE

Branching: Branching is used for executing a statement(s) or the other depending on a condition(s).

if	switch

Syntax:
`if (<condition>)`
`  <Statement1>;`
`else if (<condition>)`
`  <Statement2>;`
`  :`
`else`
`  <StatementN>;`
	Syntax:
	`switch (<Variable/Expression>)`
	`{`
	`  case <Value1>:<Statement1>;`
	`  case <Value2>:<Statement2>;`
	`  case <Value3>:<Statement3>;`
	`  default:<Statement>;`
	`}`

Allows • conditions with all relational operators • conditions with all logical operators • conditions with variables of all data types	Allows • conditions with equal to/not equal to • conditions catering to or operator • conditions with **char** and **int** variables only Note: Conditions with <,>,<=,>= will not be possible in switch

Example1: To display a message "Eligible for Class 1" if Age entered by user is equal to/more than 5 and "Not Eligible for Class 1" otherwise.

```
int Age;
cout<<"Enter Age:";
cin>>Age;
if (Age>=5)
  cout<<"Eligible for Class 1";
else
  cout<<"Not Eligible for Class 1";
```

Note: In if statement, if you want to execute more than one statement with a condition, we need to enclose them inside a set of curly braces { }. Same follows in else too.

Example2: To display grades of students based on marks (out of 100) entered by the user. Grade corresponding to marks are as follows:

```
A  Marks>=90
B  90>Marks>=75
C  75>Marks>=60
D  60>Marks>=40
E  40>Marks
float Marks;
cout<<"Marks:";cin>>Marks;
if (Marks>=90)
  cout<<"Grade A";
else if (Marks>=75)
  cout<<"Grade B";
else if (Marks>=60)
  cout<<"Grade C";
else if (Marks>=40)
  cout<<"Grade D";
else
  cout<<"Grade E";
```

Example3: To display "Upgrade Permissible" message, if user enters Trips>=10 and Month as 7 or 8.

```
int Trips, Month;
cout<<"Trips          :";cin>>Trips;
cout<<"Month(1..12):";cin>>Month;
if (Trips>=10 &&
        (Month==7 || Month==8))
  cout<<"Upgrade Permissible"<<endl;
else
  cout<<"Sorry!"<<endl;
```

Example1: To display a message "Director", "Manager", "Accountant" or "Assistant" as per the Designation codes 'D','M','A' entered by the user.

```
char Desig;
cout<<"Enter Designation Code:";
cin>>Desig;
switch (Desig)
{
 case 'D':cout<<"Director";break;
 case 'M':cout<<"Manager";break;
 case 'A':cout<<"Accountant";break;
 default:cout<<"Assistant";
}
```

Note: In the above example, just note the keyword break - it terminates the switch case after executing the statement before it.

Example2: To display the employee benefits based on designation codes entered by user. Designation Codes and corresponding benefits are as follows:

```
D-LuxuryCar,Bunglow,A.C.Cabin,Salary
M-Bunglow,A.C.Cabin,Salary
A-A.C.Cabin,Salary
Any other Code-Salary

char Desig;
cout<<"Designation Code:";
cin>>Desig;
switch (Desig)
{
 case 'D':cout<<"Lux.Car";
 case 'M':cout<<"Bunglow";
 case 'A':cout<<"A.C.Cabin";
 default:cout<<"Salary";
}
```

Note: In this example, we do not require break

Example3: To display words corresponding to alphabets.

```
char Alpha;
cout<<"Alphabets:";cin>>Alpha;
switch(Alpha)
{case 'a':
 case 'A':cout<<"Apple"<<endl;break;
 case 'b':
 case 'B':cout<<"Ball"<<endl;break;
 default:cout<<"Not known"<<endl;
}
```

Note: In this program, we take care of the uppercase as well as the lowercase alphabets.