**Experiment: 2**

PART A

(PART A: TO BE REFERRED BY STUDENTS)

**Aim:** Implementing various programs using operators, expressions and input/output operations

**Learning Outcomes: The learner would be able to**

1. Explain basic structure of C++ program
2. Identify basic data types required to solve problem statements
3. Declare variables/constants
4. Use appropriate operators and programming equivalent statements
5. Read input from console and display output to console

**Theory:**

**Simple C++ Program**

* Simple example of a C++ program that prints a string on the screen.

Printing A String “C++ is better than C”

|  |
| --- |
| #include<iostream>  Using namespace std;  int main( ) {  cout<< “C++ is better than C \n”;  return 0;  } |

Program:- 2.1 Program to print Message

This simple program demonstrates several C++ features.

**Program feature**

* Like C, the C++ program is a collection of function.
* The above example contain function **main ( ).** As usual execution begins at main ( ).
* Every C++ program must have a **main( )**.
* C++ is a free form language. With a few exception, the compiler ignore carriage return and white spaces.
* Like C, the C++ statements terminate with semicolons.

**Output Operator ( << )**

* The statement in program 2.1

cout<< “C++ is better than C.\n”;

* Causes the string in quotation marks to be displayed on the screen.
* This statement introduces two new C++ features, cout and <<.
* The identifier cout(pronounced as C out) is a predefined object that represents the standard

output stream in C++.

* Here, the standard output stream represents the screen. It is also possible to redirect the output to

other output devices.

* The operator << is called the insertion or put to operator.

**The iostream File**

* We have used the following #include directive in the program 2.1:

#include <iostream>

* The #include directive instructs the compiler to include the contents of the file enclosed within angular brackets into the source file.
* The header file **iostream** should be included at the beginning of all programs that use input/output statements.

**Return Type of main( )**

* In C++, main () returns an integer value to the operating system.
* Therefore, every main ( ) in C++ should end with a return (0) statement; otherwise a warning an error might occur.
* Since main () returns an integer type for main () is explicitly specified as **int.**
* Note that the default return type for all function in C++ is **int.**

**Input Operator ( >> )**

The statement

cin >> number1;

* Is an input statement and causes the program to wait for the user to type in a number.
* The number keyed in is placed in the variable number1.
* The identifier cin (pronounced ‘C in’) is a predefined object in C++ that corresponds to the standard input stream.
* Here, this stream represents the keyboard.
* The operator >> is known as extraction or get from operator.
* It extracts (or takes) the value from the keyboard and assigns it to the variable on its right fig 2.2.
* Like <<, the operator >> can also be overloaded.

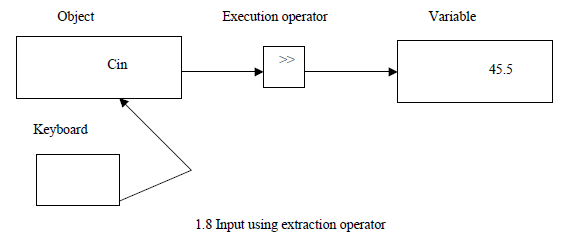


Fig:- 2.2 cin & Extraction Operator

**Cascading of input/output (I/O) Operators:**

* We have used the insertion operator << repeatedly in the last two statements for printing results.
* The statement

cout << “Sum = “ << sum << endl;

* First sends the string “Sum = “ to cout and then sends the value of sum. Finally, it sends the newline character so that the next output will be in the new line (endl).
* The multiple use of << in one statement is called cascading.
* When cascading an output operator, we should ensure necessary blank spaces between different items.
* Using the cascading technique, the last two statements can be combined as follows:

cout << “Sum = “ << sum << endl << “Average = “ << average << endl;

This is one statement but provides two line of output.

* If you want only one line of output, the statement will be:

cout << “Sum = “ << sum << “,” << “Average = “ << average << “\n”;

* + Consider sum is 14 and average is 7.

*The output will be*:

Sum = 14, average = 7

* We can also cascade input input operator >> as shown below:

cin >> number1 >> number2;

* The values are assigned from left to right. That is, if we key in two values, say, 10 and 20, then 10 will be assigned to nunber1 and 20 to number2.

**Data Types:**

* Data types or types are used to represent the type of data stored in variables.
* Data type is an instructions or keywords given to compiler for organizing data.

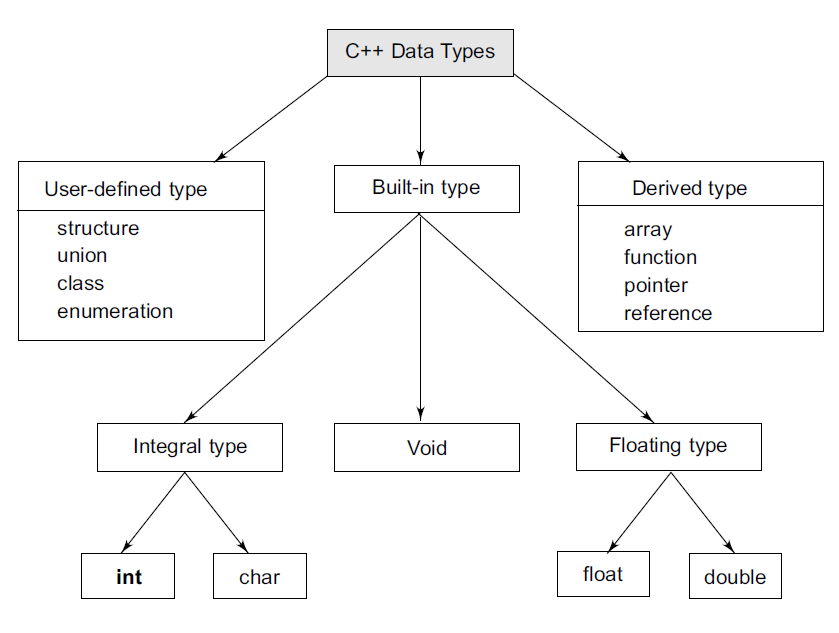
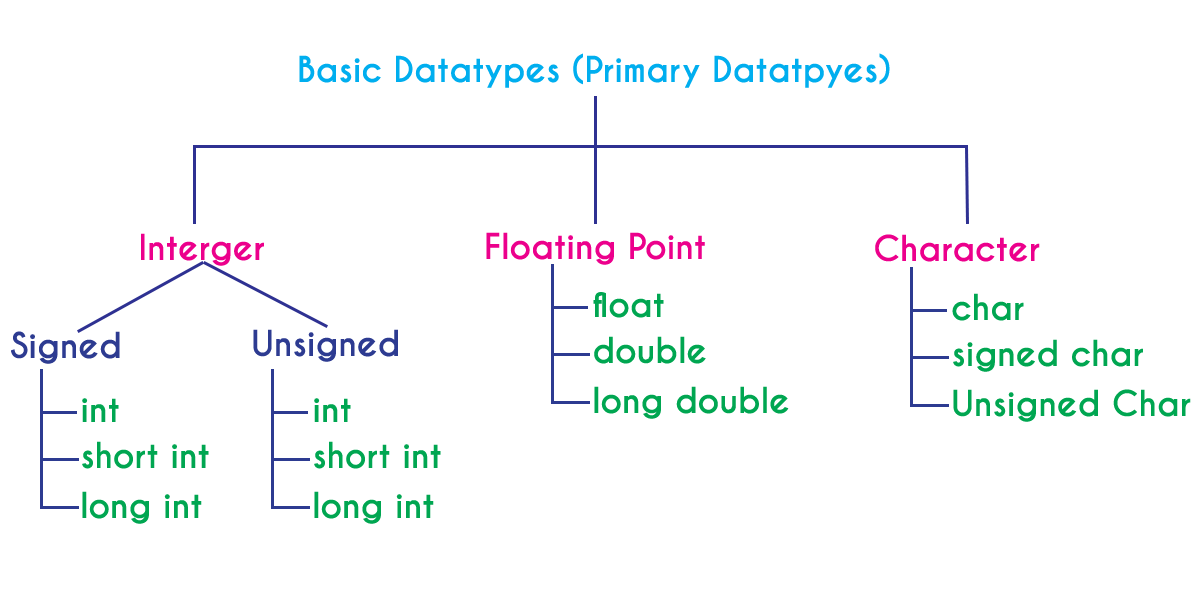


Fig: 2.1 Data Types.



* Both C and C++ compilers support all the built-in (also known as basic or fundamental or scalar or primitive) data types.

**Variable:**

* A variable is a data name that may be used to store a data value.
* Variable may take different values at different times during execution.
* A variable name can be chosen by the programmer in a meaningful way do as to reflect its function or nature in the program.
* Ex: average, amount, height , total
* Variable names may consist of letters, digits, and the underscore character.
* Variable should begin with letter. Some system permit underscore as first character.
* ANSI recognizes a length of 31 characters, however length should not be normally more than 8 characters, since only the first 8 characters are treated significant by many compilers.

**Declaration of variables:**

* After designing suitable variable names, we must declare them to the compiler. Declaration does two things:
* It tells the compiler what the variable name is.
* It specifies what type of data the variable will hold.
* Declaration of variable must be done before variable is used in program.
* General Syntax for declaring a variable:

data-type v1,v2,….,vn;

For ex: int total,age,amount;

float average;

double ratio;

short int count;

**Assigning values to the variable:**

* Values can be assigned to variable using the assignment operator = as follows:
* General syntax: variable\_name = constant value;
* For ex :
  + - * total=10;
      * weight =50;
      * balance=10.5
* C permits multiple assignments in one line:
* For ex: total=0; sum=50;
* For ex: year=year+1;
* During assignment operation, C converts the type of value on the right hand side to the type on the left.

**Constants**

* The qualifier **const** is used to create symbolic constants in C++.
* Syntax:

const *datatype vaname* **=** *value*;

* Example:

const float pi = 3.14;

**Operators in C++**

* C++ has a rich set of operators. All C operators are valid in C++ also.

|  |  |  |
| --- | --- | --- |
| ***Operators*** | ***Nature*** | ***Symbols Used*** |
| Arithmetic | Binary | +,-,\*,/,% |
| Assignment | Binary | =,+=,-=,\*=,/=,&=,>>=%=,<<= etc |
| Relational | Binary | >,<,>=,<=,==,!= |
| Logical | Binary | &&,||,! |
| Increment & Decrement | Unary | ++, -- |
| Bitwise | Binary | &, |, ~, ^, <<, >> |
| Conditional | Ternary | ?: |
| Other Special | Unary | **.**, ->, \*, &, sizeof, (type),[ ],( ),**,** |

* In addition, C++ introduces some new operators. We have already seen two such operators, namely, the insertion operator <<, and the extraction operator >>.

**Type Cast Operator**

* C++ permits explicit type conversion of variables or expressions using the type cast operator.
* Traditional C casts are augmented in C++ by a function-call notation as a syntactic alternative.
* The following two versions(Syntax) are equivalent:
* Syntax:

(type-name) expression // C notation

Type-name (expression) // C++ notation average

* examples :

Average = sum/(float)i; // C notation

Average = sum/float (i); // C++ notation

* A type-name behaves as if it is a function for converting values to a designated type.

**Expressions and Their Types**

* An expression is a combination of operators, constants and variables arranged as per the rules of the language.
* It may also include function calls which return values.
* An expression may consist of one or more operands, and zero or more operators to produce a value.
* Expressions may be of the following seven types:
  + Constant expressions (20+30)
  + Integral expressions (a+b)
  + Float expressions (3.147f\*r\*r)
  + Pointer expressions(&ref)
  + Relational expressions(a>b)
  + Logical expressions(a>b&&a>c)
  + Bitwise expressions(a>>1)
* An expression may also use combinations of the above expressions. Such expressions are known as compound expressions.

**Precedence of Operator or Priority.**

|  |  |  |
| --- | --- | --- |
| **Precedence** | **Operator Symbol** | **Associativity of Operator** |
| 0. | :: | Left to Right |
| 1. | ( ) [ ] . -> | Left to Right |
| 2. | Unary +, Unary -, ++, --, ~, ! ,\*, &, sizeof, (type), new , delete | Right to Left |
| 3. | \* / % | Left to Right |
| 4. | + - |
| 5. | << >> |
| 6. | <<= >>= |
| 7. | == != |
| 8. | & |
| 9. | ^ |
| 10. | | |
| 11. | && |
| 12. | || |
| 13. | ?: | Right to Left |
| 14. | = += -= \*= /= %= | Right to Left |
| 15. | , | Left to Right. |

Example: Evaluate the expression 10 + 4 \* 3 / 2

In the above expression there are three operators **+, \* and /**. Among these three operators, both multiplication and division have same higher precedence and addition has lower precedence. So, according to the operator precedence both multiplication and division are evaluated first and then addition is evaluated. As multiplication and division have same precedence they are evaluated based on the associativity. Here, the associativity of multiplication and division is **left to right**. So, multiplication is performed first, then division and finally addition. So, the above expression is evaluated in the order of **\* / and +**. It is evaluated as follows...

(10 + ((4 \* 3) / 2))

4 \* 3 ====> 12  
12 / 2 ===> 6  
10 + 6 ===> 16  
The expression is evaluated to **16**.

**Tasks:**

|  |  |
| --- | --- |
|  | Write a program to initialize your details like age, name, gender, city, height etc and display it. (for name & city use character array ex. char name [20]) |
|  | Write a program to read your details like age, name, gender, city, height etc and display it. |
|  | Write a program to exchange values of two variables without using 3rd variable |
|  | Given the value of x, y, and z. Write a program to rotate their values such that x has value of y, y has value of z and z has value of x. |
|  | Write a program to find area & perimeter of a circle |
|  | Write a program to calculate simple interest. |
|  | Write a program to convert temperature in Celsius to Fahrenheit. |
|  | A four-digit number is inputted through the keyboard. Write a program to calculate sum of digits of a number. |
|  | A four-digit number is inputted through the keyboard. Write a program to reverse the number. |
|  | Write a program to find largest of two numbers using ternary operator. |
|  | If the length of three sides of a triangle is input through the keyboard, write a program to find the area of triangle and check whether the triangle is valid or not using conditional operator. Hint: - A triangle is valid if the sum of its two sides is greater than the third side. |
|  | Write a program to calculate compound interest. |

**Question of Curiosity**

[To be answered by student based on the practical performed and learning/observations]

1. Convert Following Mathematical Equations to programming equivalent statement
2. r =
3. What is operator? List various Operators.
4. What is variable and constant? Explain.
5. What are rules for naming the variables/identifiers?
6. What is precedence of Operators?

**Tasks:**

**Q1.**

**Code:**

// Write a program to initialize your details like age, name, gender, city, height etc and display it. (for name & city use character array ex. char name [20])

#include <iostream>

using namespace std;

int main()

{

char name[25] = "Manan";

char city[25] = "Mumbai";

unsigned short int age = 17;

char gender[10] = "Male";

unsigned int height = 180;

cout << "Age: " << age << " Years" << endl;

cout << "Name: " << name << endl;

cout << "Gender: " << gender << endl;

cout << "City: " << city << endl;

cout << "Height: " << height << "cm" << endl;

return 0;

}

**Output:**

A blue background with white text

Description automatically generated

**Q2.**

**Code:**

// Write a program to read your details like age, name, gender, city, height etc and display it.

#include <iostream>

using namespace std;

int main()

{

unsigned short int age = 0;

char name[25] = "";

char gender[10] = "";

char city[25] = "";

unsigned int height = 100;

cout << "Enter age: ";

cin >> age;

cout << "Enter name: ";

cin >> name;

cout << "Enter gender: ";

cin >> gender;

cout << "Enter city: ";

cin >> city;

cout << "Enter height: ";

cin >> height;

cout << "Age: " << age << " Years" << endl;

cout << "Name: " << name << endl;

cout << "Gender: " << gender << endl;

cout << "City: " << city << endl;

cout << "Height: " << height << "cm" << endl;

return 0;

}

**Output:**

**A screen shot of a computer

Description automatically generated**

**Q3.**

**Code:**

// Write a program to exchange values of two variables without using 3rd variable

#include <iostream>

using namespace std;

int main()

{

int n1 = 0, n2 = 0;

cout << "Enter n1: ";

cin >> n1;

cout << "Enter n2: ";

cin >> n2;

n2 = n1 + n2;

n1 = n2 - n1;

n2 = n2 - n1;

cout << "n1: " << n1 << endl;

cout << "n2: " << n2 << endl;

return 0;

}

**Output:**

**A blue background with white text

Description automatically generated**

**Q4.**

**Code:**

// Given the value of x, y, and z. Write a program to rotate their values such that x has value of y, y has value of z and z has value of x.

#include <iostream>

using namespace std;

int main()

{

int x = 0, y = 0, z = 0;

cout << "Enter x: ";

cin >> x;

cout << "Enter y: ";

cin >> y;

cout << "Enter z: ";

cin >> z;

z = x + y + z;

y = z - x - y;

x = z - x - y;

z = z - x - y;

cout << "x: " << x << endl;

cout << "y: " << y << endl;

cout << "z: " << z << endl;

return 0;

}

**Output:**

**A screen shot of a computer

Description automatically generated**

**Q5.**

**Code:**

// Write a program to find area & perimeter of a circle

#include <iostream>

using namespace std;

int main() {

unsigned int radius = 0;

double area, perimeter;

cout << "Enter radius: ";

cin >> radius;

area = 3.14 \* radius \* radius;

perimeter = 2 \* 3.14 \* radius;

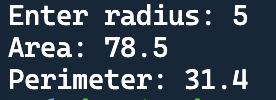
cout << "Area: " << area << endl;

cout << "Perimeter: " << perimeter << endl;

return 0;

}

**Output:**

****

**Q6.**

**Code:**

// Write a program to calculate simple interest

#include <iostream>

using namespace std;

int main()

{

double p = 0, r = 7, t = 2;

float interest = 0, amount = 0;

cout << "Enter principal: ";

cin >> p;

cout << "Enter rate of interest: ";

cin >> r;

cout << "Enter time in years: ";

cin >> t;

interest = p \* r \* t / 100;

amount = interest + p;

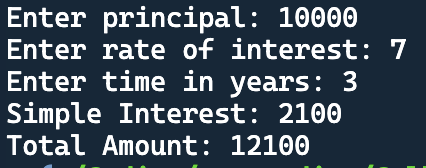
cout << "Simple Interest: " << interest << endl;

cout << "Total Amount: " << amount << endl;

return 0;

}

**Output:**

****

**Q7.**

**Code:**

// Write a program to convert temperature in Celsius to Fahrenheit.

#include <iostream>

using namespace std;

int main()

{

int celsius = 0, fahrenheit = 0;

cout << "Enter temperature in Celsius: ";

cin >> celsius;

fahrenheit = (celsius \* 9/5) + 32;

cout << "Temperature in Fahrenheit: " << fahrenheit << endl;

return 0;

}

**Output:**

A close up of a sign

Description automatically generated

**Q8.**

**Code:**

// A four-digit number is inputted through the keyboard. Write a program to calculate sum of digits of a number.

#include <iostream>

using namespace std;

int main()

{

int n = 0, sum = 0;

start:

cout << "Enter a 4 digit number: ";

cin >> n;

if (n < 1000 || n > 9999)

{

cout << n << " is not a 4 digit number" << endl;

goto start;

}

while (n > 0)

{

sum += n % 10;

n /= 10;

}

cout << "Sum of digits: " << sum << endl;

return 0;

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**Q9.**

**Code:**

// A four-digit number is inputted through the keyboard. Write a program to reverse the number.

#include <iostream>

using namespace std;

int main()

{

int n = 0, reverse = 0;

start:

cout << "Enter a 4 digit number: ";

cin >> n;

if (n < 1000 || n > 9999)

{

cout << n << " is not a 4 digit number" << endl;

goto start;

}

while (n > 0)

{

reverse = (reverse \* 10) + (n % 10);

n /= 10;

}

cout << "Reverse: " << reverse << endl;

return 0;

}

**Output:**

**A blue background with white text

Description automatically generated**

**Q10.**

**Code:**

// Write a program to find largest of two numbers using ternary operator.

#include <iostream>

using namespace std;

int main()

{

int a, b, largest;

cout << "Enter a: ";

cin >> a;

cout << "Enter b: ";

cin >> b;

largest = a > b ? a : b;

cout << "Largest: " << largest << endl;

return 0;

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**Q11.**

**Code:**

// If the length of three sides of a triangle is input through the keyboard, write a program to find the area of triangle and check whether the triangle is valid or not using conditional operator. Hint: - A triangle is valid if the sum of its two sides is greater than the third side.

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double a, b, c, area, semi\_perimeter;

cout << "Enter a: ";

cin >> a;

cout << "Enter b: ";

cin >> b;

cout << "Enter c: ";

cin >> c;

if (((a + b) > c) && ((b + c) > a) && ((c + a) > b))

{

semi\_perimeter = (a + b + c) / 2;

area = sqrt(semi\_perimeter \* (semi\_perimeter - a) \* (semi\_perimeter - b) \* (semi\_perimeter - c));

cout << "Area: " << area << endl;

}

else

{

cout << "Triangle is not valid" << endl;

}

return 0;

}

**Output:**

**A number on a blue background

Description automatically generated**

**Q12.**

**Code:**

// Write a program to calculate compound interest.

#include <iostream>

using namespace std;

int main()

{

double p, r, t, amount, interest;

cout << "Enter principal: ";

cin >> p;

cout << "Enter rate of interest: ";

cin >> r;

cout << "Enter time in years: ";

cin >> t;

amount = p \* pow((1 + (r / 100)), t);

interest = amount - p;

cout << "Compound Interest: " << interest << endl;

cout << "Amount: " << amount << endl;

return 0;

}

**Output:**

**A blue background with white text

Description automatically generated**