INTRODUCTION TO C++

Unit I – Chapter 2

Topics to be covered…

- History of C++
- Structure of C++
- Application of C++
- Tokens
- Data Types
  - Basic data types
  - Derived data types
- Constants
  - Symbolic Constants
- Dynamic initialization
- Variables,
- Scope resolution operator,
- Type modifiers,
- Type-casting operators and control statements
- Input and Output statements in C++
- Function prototyping and components,
- Passing-parameters:
  - Call by reference
  - Return by reference
- Inline function
- Default arguments
- Overloaded function.

2.1 History of C++

- The C++ language is an ***object-oriented programming language***.
- It is a combination of both low-level & high-level language → ***Middle-Level Language.***
- The programming language was created, designed & developed by a Danish Computer Scientist – ***Bjarne Stroustrup*** at Bell Telephone Laboratories (now known as Nokia Bell Labs) in Murray Hill, New Jersey.
- As he wanted a flexible & a dynamic language which was similar to C with all its features, but with additionality of active type checking, basic inheritance, default functioning argument, classes, etc. and hence C with Classes (C++) was launched

2.1 History of C++ ….

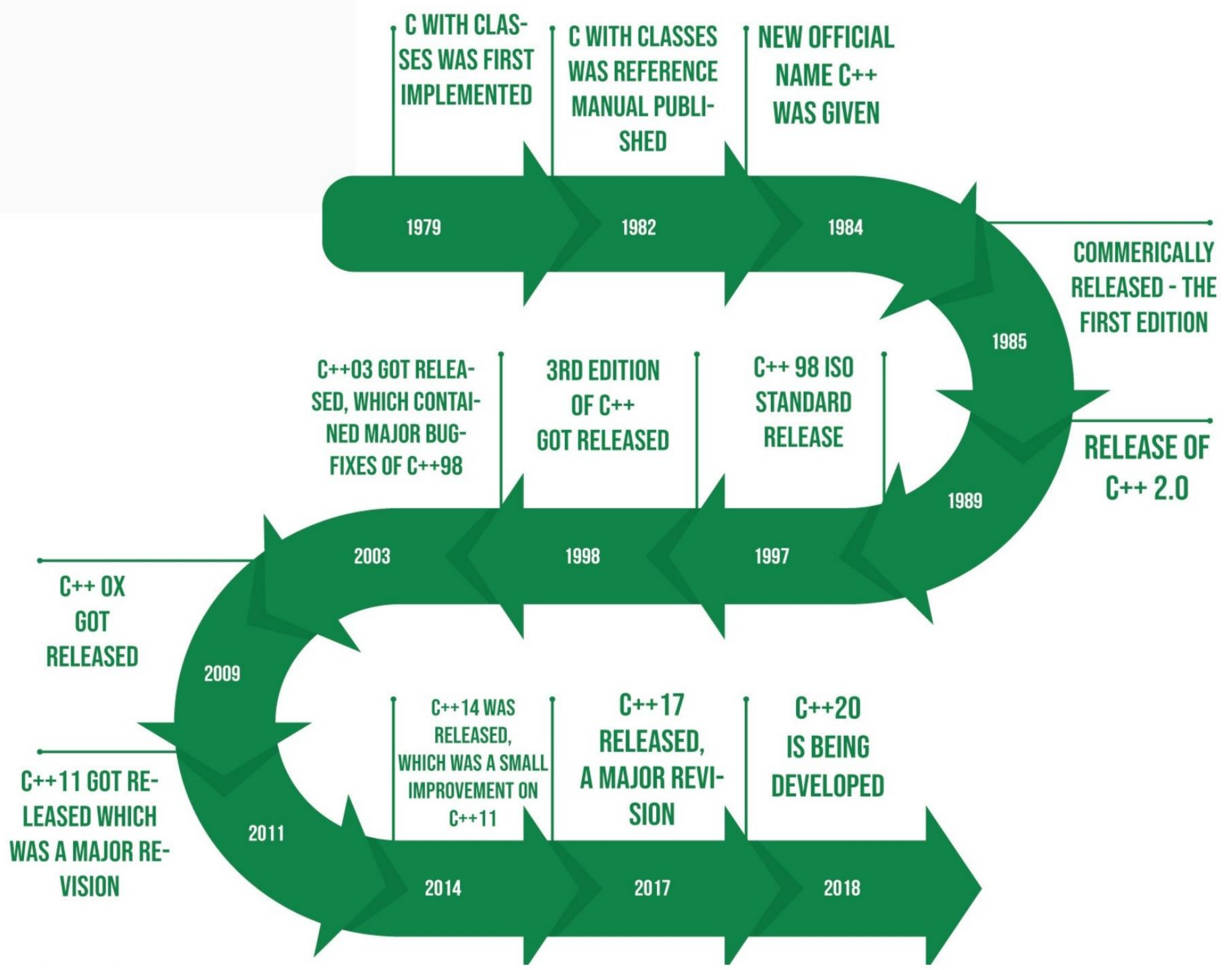
- C++ was initially known as "C with classes, " and was renamed C++ in 1983. ++ is shorthand for adding one to variety in programming; therefore C++ roughly means that "one higher than C"
- **Bjarne Stroustrup** is known as the founder of C++ language.

2.1 History of C++ ….

- It was develop for adding a feature of OOP (Object Oriented Programming) in C without significantly changing the C component.
- C++ programming is "relative" (called a superset) of C, it means any valid C program is also a valid C++ program.

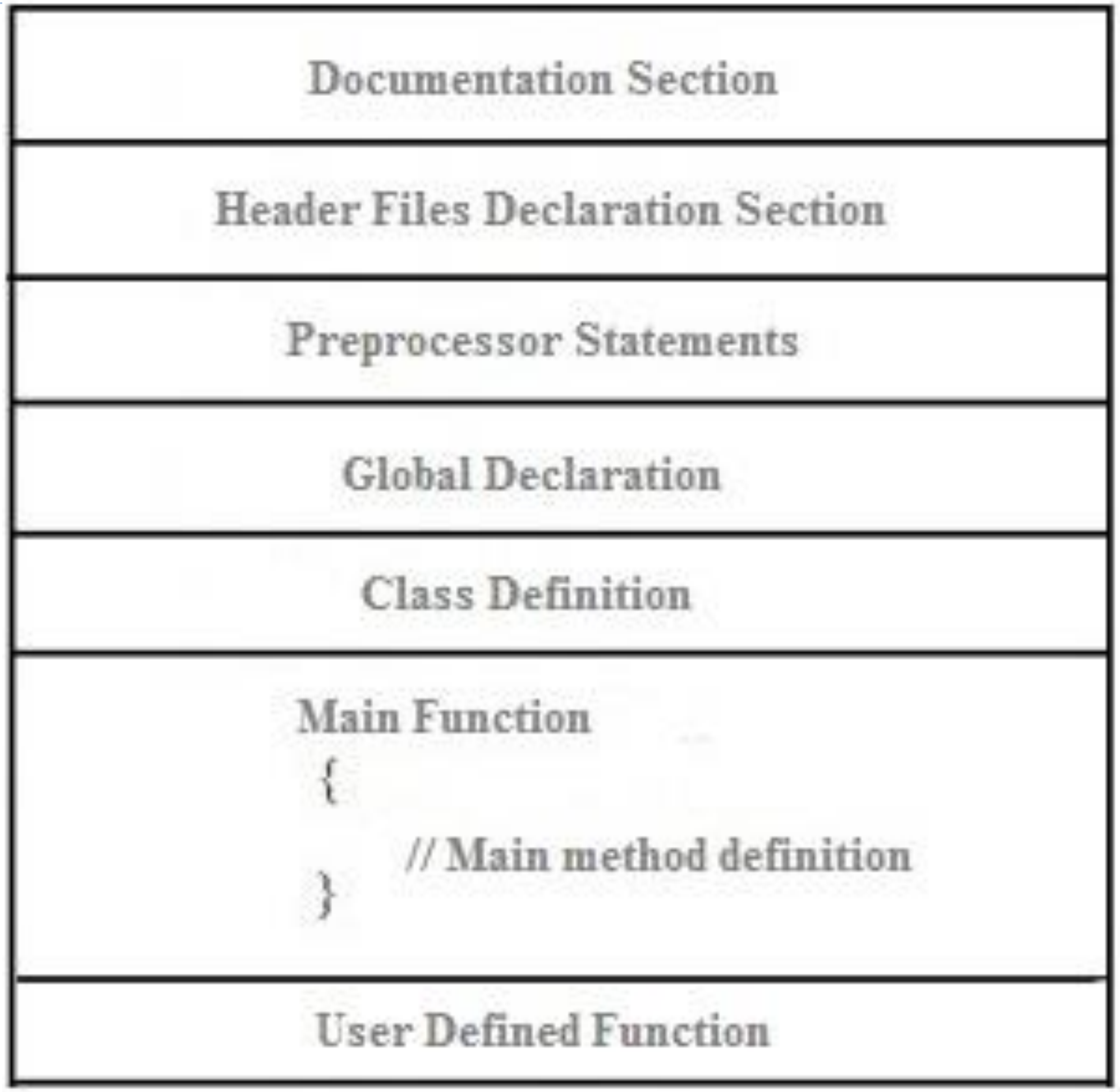
Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
K & R C	1978	Kernighan & Dennis Ritchie
C++	1980	Bjarne Stroustrup

History of C++


C WITH CLASSES WAS FIRST IMPLEMENTED
1979
C WITH CLASSES WAS REFERENCE MANUAL PUBLISHED
1982
NEW OFFICIAL NAME C++ WAS GIVEN
1984
COMMERICALLY RELEASED - THE FIRST EDITION
1985
RELEASE OF C++ 2.0
1989
C++ 98 ISO STANDARD RELEASE
1997
3RD EDITION OF C++ GOT RELEASED
1998
C++03 GOT RELEASED, WHICH CONTAINED MAJOR BUG-FIXES OF C++98
2003
C++ 0X GOT RELEASED
2009
C++11 GOT RELEASED WHICH WAS A MAJOR REVISION
2011
C++14 WAS RELEASED, WHICH WAS A SMALL IMPROVEMENT ON C++11
2014
C++17 RELEASED, A MAJOR REVISION
2017
C++20 IS BEING DEVELOPED
2018

YENEPOYA (DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

2.2 Program Structure of C++



YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

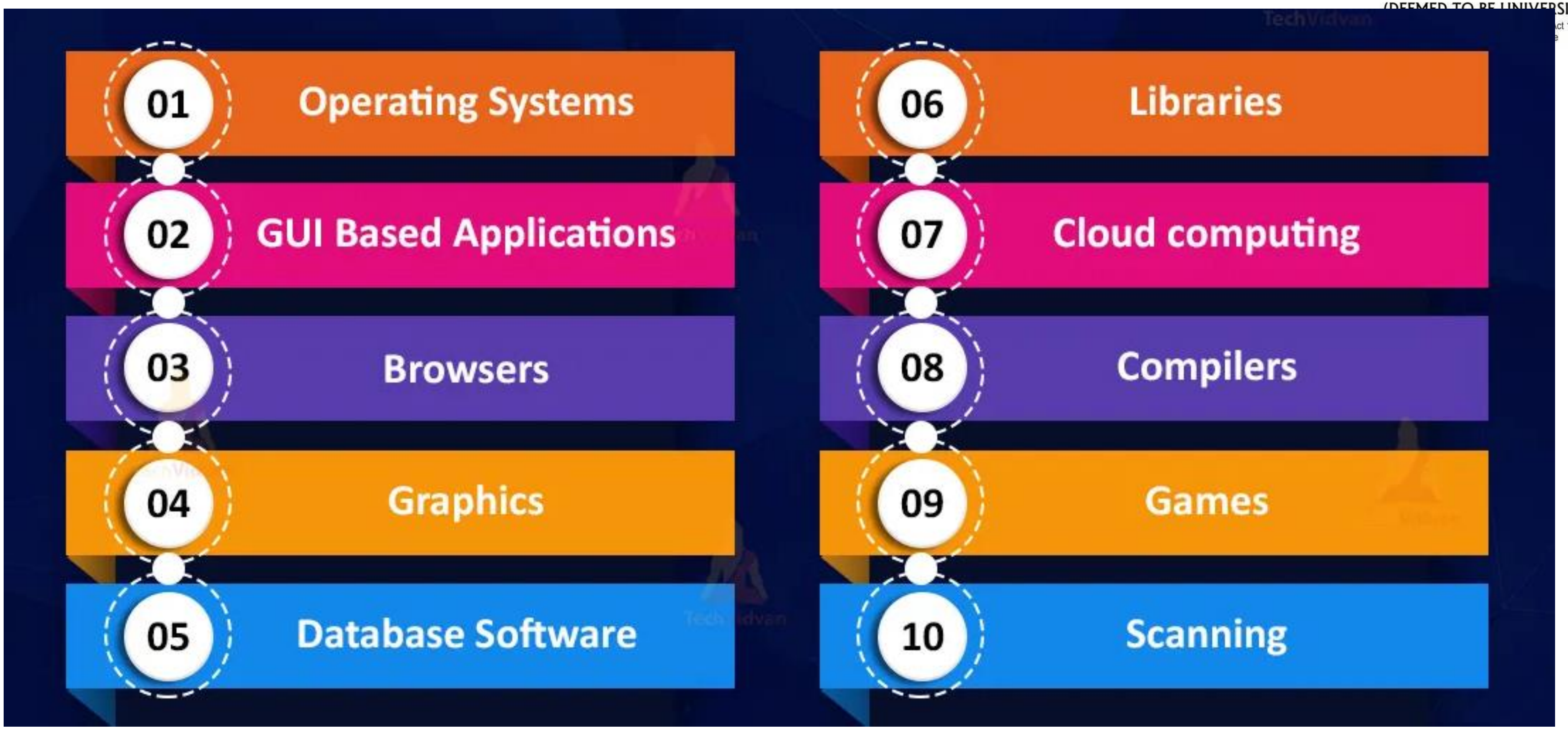
Example:

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

```
/*
* File: demo.cpp
* Author: BCABigData
*/
#include <iostream.h>
int main()
{
   cout<<"This is my first C++ Program";
   cout<<endl<<"and its very easy";
   return (0);
}
```

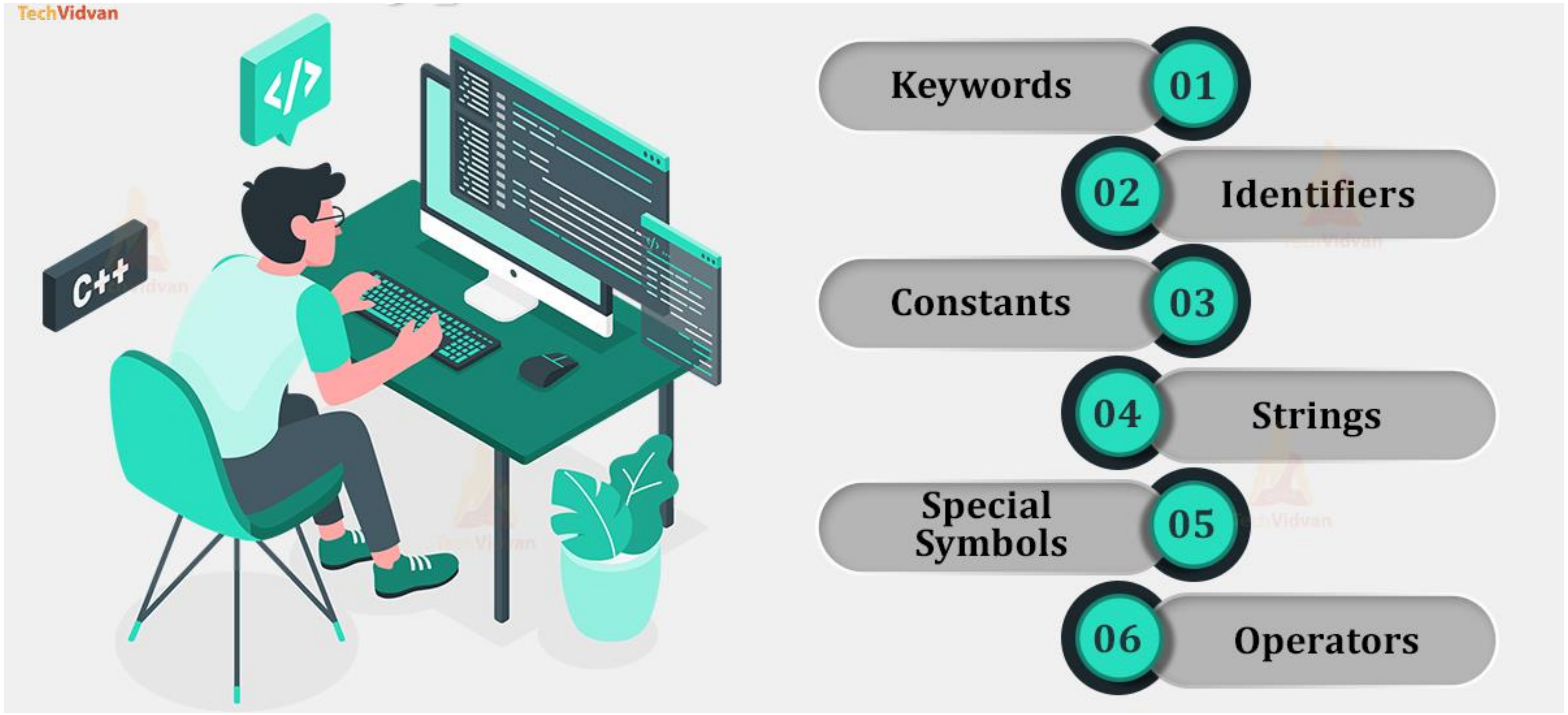
2.3 Application of C++

YENEPOYA
(DEEMED TO BE UNIVERSITY)


01 Operating Systems
02 GUI Based Applications
03 Browsers
04 Graphics
05 Database Software
06 Libraries
07 Cloud computing
08 Compilers
09 Games
10 Scanning

2.4 Tokens

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade



01 Keywords
02 Identifiers
03 Constants
04 Strings
05 Special Symbols
06 Operators

2.4.1 Keywords

- Keywords are ***pre-defined*** or ***reserved words*** in a programming language.
- Each keyword is meant ***to perform a specific function*** in a program.
- Since keywords are referred names for a compiler, they can't be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed.
- ***Cannot redefine*** keywords.
- However, you can specify the text to be substituted for keywords before compilation by using C/C++ preprocessor directives.

C- 32

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

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

and additional 31 of C++

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

2.4.2 Identifiers

- Identifiers are used as the general terminology for the naming of variables, functions and arrays.
- These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character.
- Identifier names must differ in spelling and case from any keywords.
- Cannot use keywords as identifiers; they are reserved for special use.
- Once declared, we can use the identifier in later program statements to refer to the associated value.
- A special kind of identifier, called a statement label, can be used in goto statements.

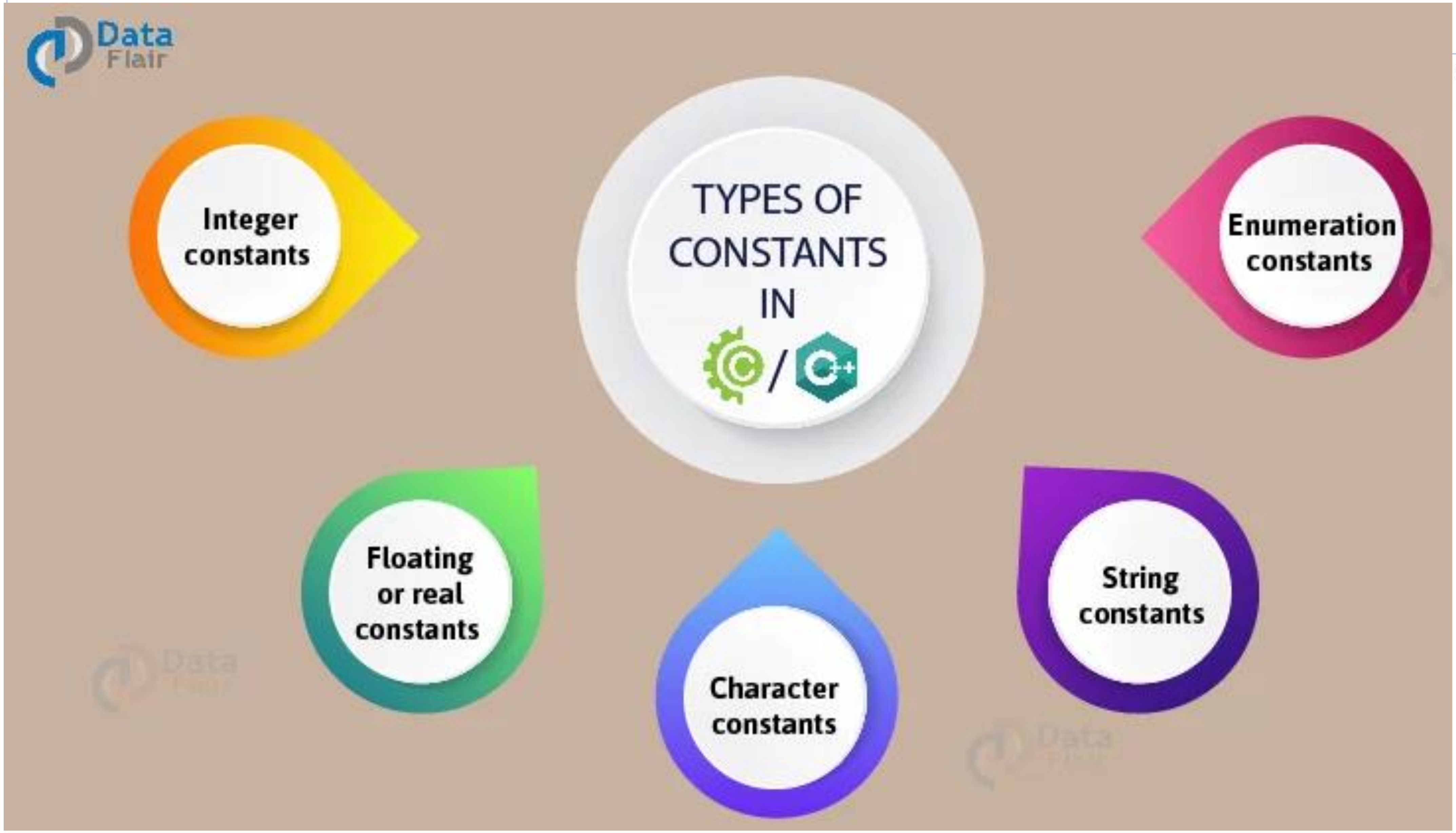
Rules for naming a Identifier

- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore.
- No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only the first 31 characters are significant.

2.4.3 Constants

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- Constants are also like normal variables.
- But, the only difference is, their values can not be modified by the program once they are defined.
- Constants refer to fixed values.
- They are also called literals.
- Constants may belong to any of the data type


Data Flair
TYPES OF CONSTANTS IN C / C++
Integer constants
Enumeration constants
Floating or real constants
Character constants
String constants

2.4.3.1 Integer Constants

- An integer constant is an integer with a fixed value, that is, it cannot have fractional value like 10, -8, 2019.
- For example: const signed int limit = 20;
- Classify it into three types, namely:
- **Decimal number system constant:**
  - It has the base/radix 10. (0 to 9)
  - For example, 55, -20, 1.
  - In the decimal number system, no prefix is used.
- **Octal number system constant:**
  - It has the base/radix 8. (0 to 7)
  - For example, 034, 087, 011.
  - In the octal number system, 0 is used as the prefix.
- **Hexadecimal number system constant:**
  - It has the base/radix 16. (0 to 9, A to F)
  - In the hexadecimal number system, 0x is used as the prefix.
  - C language gives you the provision to use either uppercase or lowercase alphabets to represent hexadecimal numbers.

2.4.3.2 Floating or Real Constants

- Represent all the real numbers on the number line, which includes all fractional values.
- For instance: const long float pi = 3.14159;
- Represent it in 2 ways:
- **Decimal form:**
  - The inclusion of the decimal point (.) is mandatory.
  - For example, 2.0, 5.98, -7.23.
- **Exponential form:**
  - The inclusion of the signed exponent (either e or E) is mandatory.
  - For example, the universal gravitational constant G = 6.67 x 10-11 is represented as 6.67e-11 or 6.67E-11.

2.4.3.3 Character Constants

- Character constants are used to assign a fixed value to characters including alphabets and digits or special symbols
- They are enclosed within ***single quotation marks***(‘ ’).
- Each character is associated with its specific numerical value called the ASCII (American Standard Code For Information Interchange) value.
- For example, ‘+’, ‘A’, ‘d’.

2.4.3.4 Strings

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- Strings are nothing but an array of characters ended with a null character ('\0').
- This null character indicates the end of the string.
- Strings are always enclosed in ***double-quotes.***
- Whereas, a character is enclosed in single quotes in C and C++
- char string[20] = {'h','e','l','l','o' '\0'};
- char string[20] = "hello";
- char string [] = "hello";
- When we declare char as "string[20]", 20 bytes of memory space is allocated for holding the string value.
- When we declare char as "string[]", memory space will be allocated as per the requirement during the execution of the program.

2.4.3.5 Special Symbols

- **Brackets[]:** Opening and closing brackets are used as array element reference.
  - These indicate single and multidimensional subscripts.
- **Parentheses():** These special symbols are used to indicate function calls and function parameters.
- **Braces{}:** These opening and ending curly braces mark the start and end of a block of code containing more than one executable statement.
- **Comma (,):** It is used to separate more than one statements like for separating parameters in function calls.

2.4.3.5 Special Symbols….

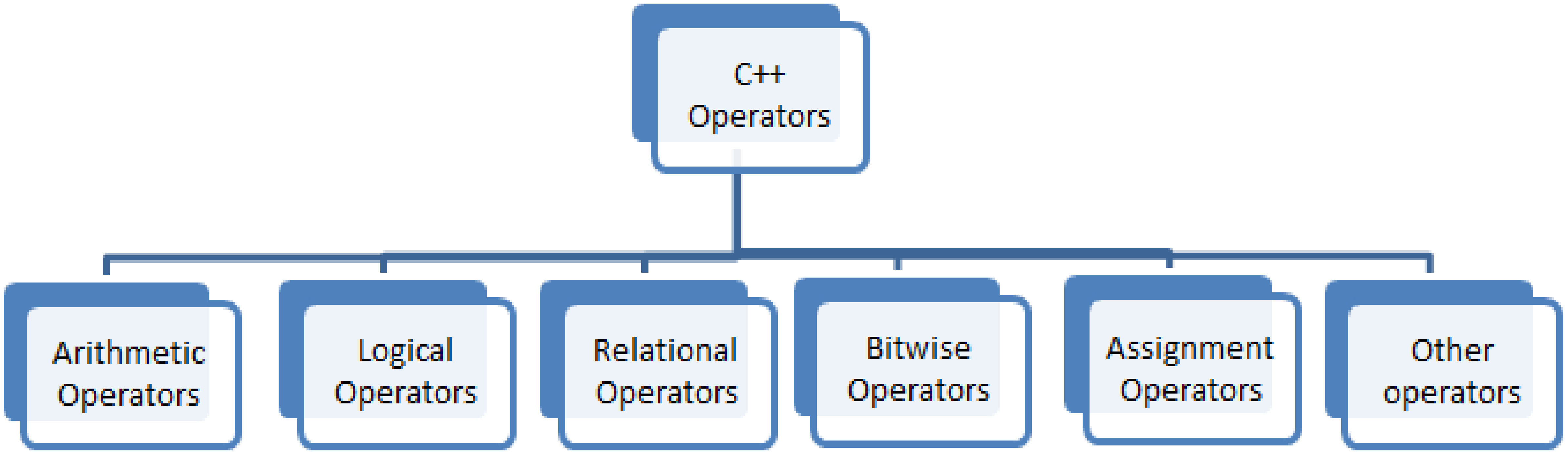
YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- **Colon(:):** It is an operator that essentially invokes something called an initialization list.
- **Semicolon(;):** It is known as a *statement terminator*.
  - It indicates the end of one logical entity.
  - That's why each individual statement must be ended with a semicolon.
- **Asterisk (*):** It is used to create a pointer variable.
- **Assignment operator(=):** It is used to assign values.
- **Pre-processor (#):** The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.

2.4.3.6 Operators

- Operators are symbols that trigger an action when applied to C variables and other objects.
- The data items on which operators act upon are called operands.
- **Unary Operators:**
- Which require only a single operand to act upon are known as *unary operators*.
- – increment and decrement operators
- **Binary Operators:**
- Which operators that require two operands to act upon are called *binary operators*.

• **Binary operators** are classified into :
– Arithmetic operators
– Relational Operators
– Logical Operators
– Assignment Operators
– Conditional Operators
– Bitwise Operators


C++ Operators
Arithmetic Operators
Logical Operators
Relational Operators
Bitwise Operators
Assignment Operators
Other operators

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Relational Operators

Operator	Description
==	Is equal to
!=	Is not equal to
>	Greater than
<	Less
>=	Greater than or equal to
<=	Less than or equal to

Logical Operators

&&	And operator. Performs logical conjunction of two expressions.(if both expressions evaluate to True, result is True. If either expression evaluates to False, the result is False) x < 5 && x < 10
||	Or operator. Performs a logical disjunction on two expressions.(if either or both expressions evaluate to True, the result is True) x < 5 || x < 4
!	Not operator. Performs logical negation on an expression. !(x < 5 && x < 10)

Bitwise Operators

<<	Binary Left Shift Operator
!=	Is not equal to
>>	Binary Right Shift Operator
~	Binary One's Complement Operator
&	Binary AND Operator
^	Binary XOR Operator
|	Binary OR Operator

Types of Bitwise Operators

Operator	Name	Example	Result
&	Bitwise AND	6 & 3	2
|	Bitwise OR	10 | 10	10
^	Bitwise XOR	2^2	0
~	Bitwise 1's complement	~9	-10
<<	Left-Shift	10<<2	40
>>	Right-Shift	10>>2	2

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

Bitwise Operator

- ✓ Applied to integer type – long, int, short, byte and char.

A	B	A | B	A & B	A^ B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

Assignment

=	Assign
+=	Increments, then assign
-=	Decrements, then assign
*=	Multiplies, then assign
/=	Divides, then assign
%=	Modulus, then assigns
<<=	Left shift and assigns
>>=	Right shift and assigns
&=	Bitwise AND assigns
^=	Bitwise exclusive OR and assigns
|=	Bitwise inclusive OR and assigns

Miscellaneous Operators

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

,	Comma operator
sizeOf()	Returns the size of a memory location.
&	Returns the address of a memory location.
*	Pointer to a variable.
? :	Conditional Expression –ternary

2.4.3.7 Variables

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

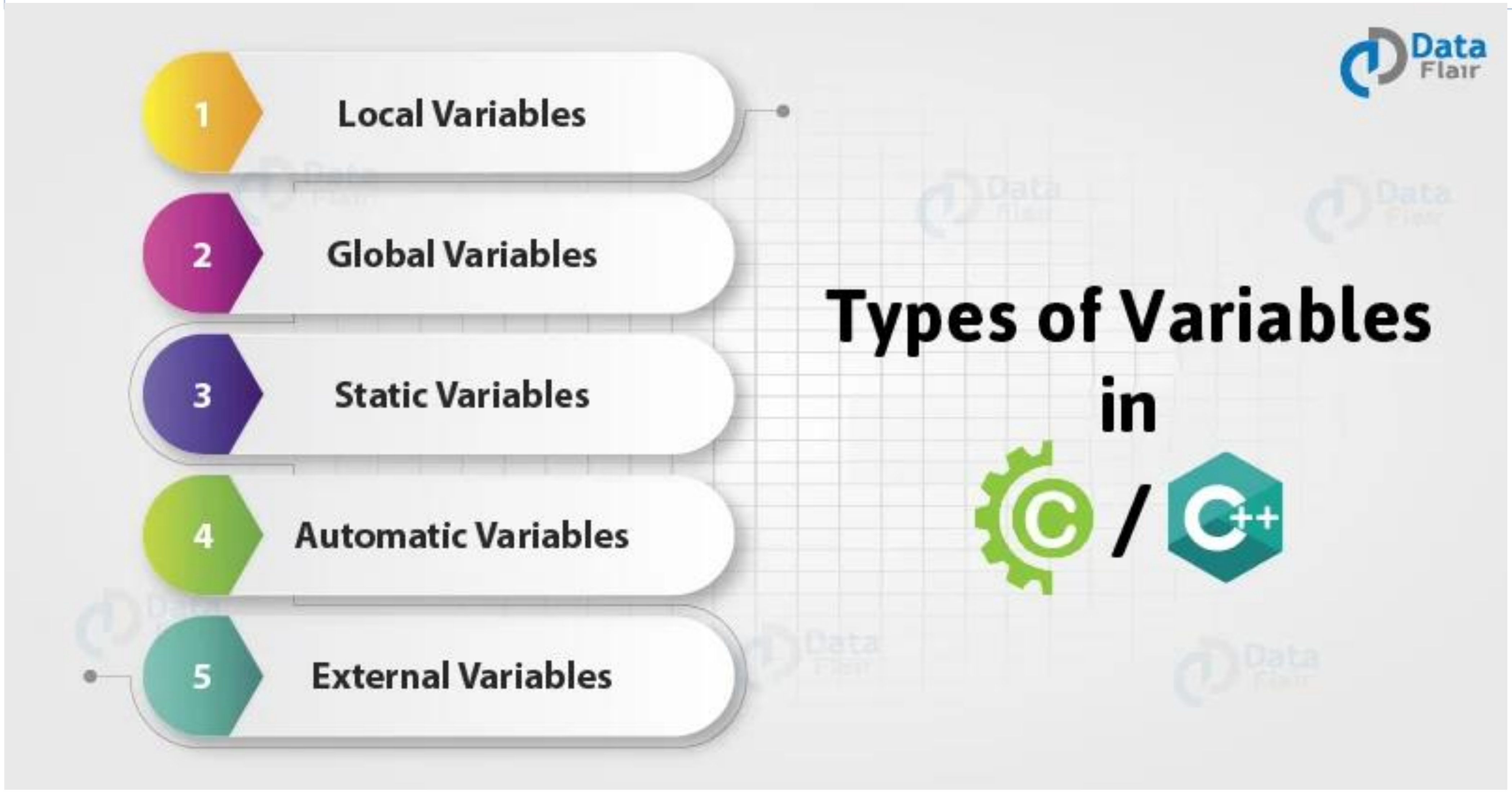
- A variable in simple terms is a storage place which has some memory allocated to it.
- Basically, a variable used to store some form of data.
- Different types of variables require different amounts of memory, and have some specific set of operations which can be applied on them.
- datatype variable_name;
- datatype variable1_name, variable2_name, variable3_name;
- int num;
- num=10;
- int num1=10,num2=20,num3;
- int a=10,b=20,sum;

Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only.
- It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword,
- e.g. int, goto , etc.

```
int student_id;
```

- Variable declaration refers to the part where a variable is first declared or introduced before its first use.
- Variable definition is the part where the variable is assigned a memory location and a value


Data Flair
1 Local Variables
2 Global Variables
3 Static Variables
4 Automatic Variables
5 External Variables
Types of Variables in C / C++

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- **1. Local Variables**
- The scope of local variables lies only within the function or the block of code.
- These variables stay in the memory till the end of the program.
- Local variables need to be initialized before use

- **2. Global Variables**
- A global variable has a global scope, that is, this variable is valid until the end of the program.
- It is declared at the starting of program.
- These variables are available to all the functions in that program.

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- **3. Static Variables**
- Static variable retains its value within the function calls. 'static' keyword is used to define a static variable.
- A static variable can preserve its value, and you can not initialize a static variable again.
- With every function call, static variable uses the preserved value and increment the value of a variable, whereas a local variable re-initializes the value every time function calls takes place.

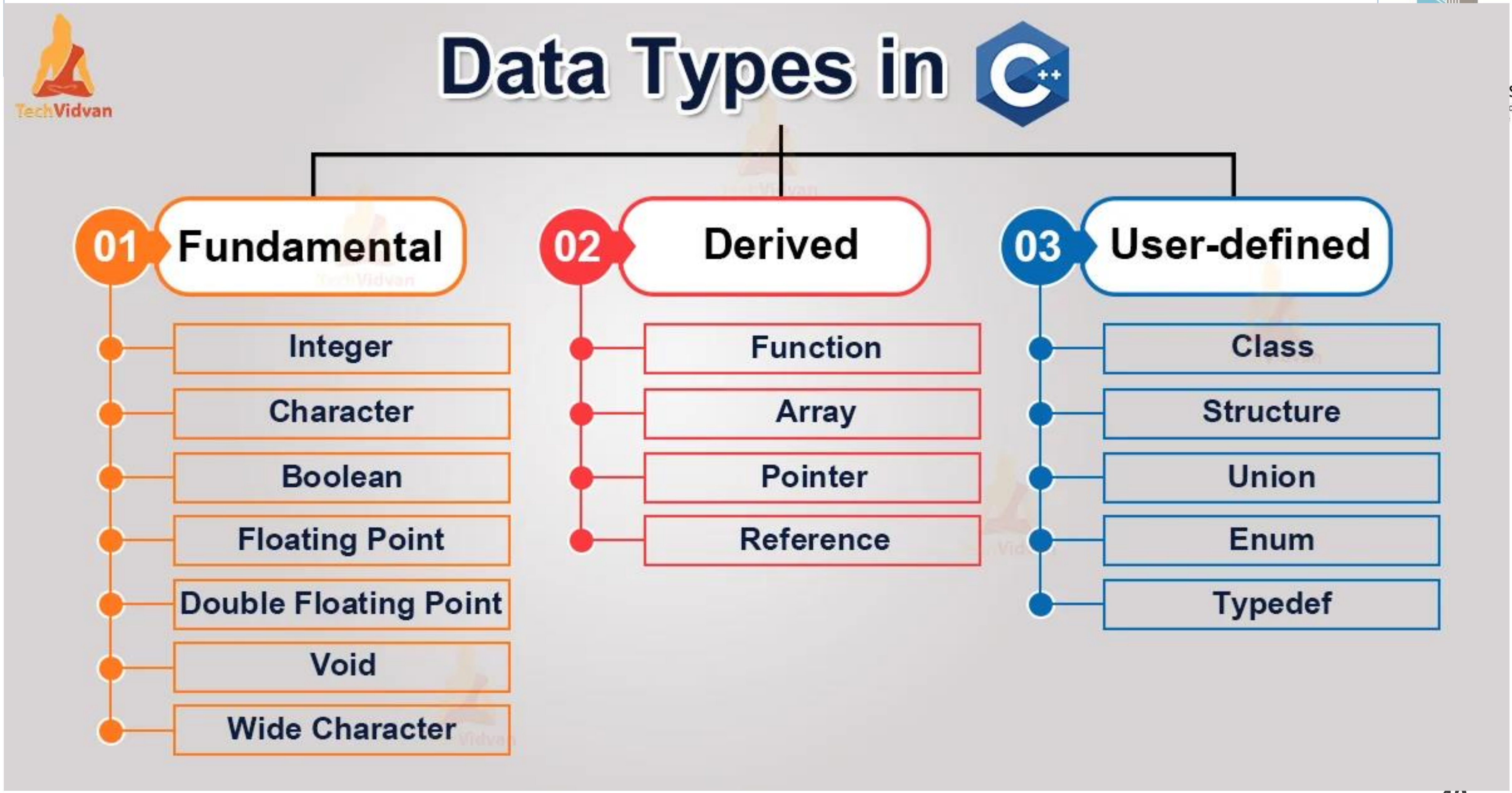
```
#include <iostream>

using namespace std;

void statf();

int main ()

{
                        void statf()

 int i;
{

            int a = 20; //local variable

                        static int b = 20; //static variable

for(i = 0; i < 5; i++)                                                                a = a + 1;

  {
                        b = b + 1;

    statf();
cout<<"Value of a: " << a;

  }
                        cout<<“\n Value of b: "<< b;

}
                        }
```

- **4. Automatic Variables**
- The variable declared inside a block is called an *automatic variable.*
- The automatic variables in C are different from the local variables,
  - automatic variable allocates memory upon the entry to that block and frees the occupied space when the control exits from the block.
- ‘auto’ keyword defines an automatic variable.
- Example: auto int var = 39;

- **5. External Variables**
- **‘extern’** keyword to increase the visibility of the program.
- An extern variable is available to other files too.
- The difference between global and extern variable is, a global variable is available anywhere within the file or program in which global variable declares but an extern variable is available for other files too.
- Note: Variables with keyword ‘extern’ are only declared; they are not defined. Also, initialization of extern keywords can also be considered as its definition.
- Example: extern int var;

2.5 C++ Data Types

- While writing a program, various variables are used to store various information.
- Variables are nothing but reserved memory locations to store values.
- This means that when a variable is created you reserve some space in memory.
- You may like to store information of various data types like character, wide character, integer, floating point, double floating point, Boolean, etc.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.


TechVidvan
Data Types in C++
01 Fundamental
Integer
Character
Boolean
Floating Point
Double Floating Point
Void
Wide Character
02 Derived
Function
Array
Pointer
Reference
03 User-defined
Class
Structure
Union
Enum
Typedef

SITY)
ct 1956

- Some basic types can be modified using one or more of
- these type modifiers –
- – signed
- – unsigned
- – short
- – long
- The modifiers signed, unsigned, long, and short can be applied to integer base types.
- In addition, signed and unsigned can be applied to char, and long can be applied to double.
- The modifiers signed and unsigned can also be used as prefix to long or short modifiers. For example, unsigned long int.
- unsigned x;
- unsigned int y;

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-2,147,483,648 to 2,147,483,647
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 4,294,967,295
long long int	8bytes	-(2^63) to (2^63)-1
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character

Check your system data size

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

```
#include <iostream.h>
void main()
{
cout << "Size of char : " << sizeof(char) << endl;
cout << "Size of int : " << sizeof(int) << endl;
cout << "Size of short int : " << sizeof(short int) << endl;
cout << "Size of long int : " << sizeof(long int) << endl;
cout << "Size of float : " << sizeof(float) << endl;
cout << "Size of double : " << sizeof(double) << endl;
cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;
}
```

2.6 C++ Variable Types

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- A variable provides us with named storage that our programs can manipulate.
- Each variable in C++ has a specific type, which determines
  - the size and layout of the variable's memory;
  - the range of values that can be stored within that memory; and
  - the set of operations that can be applied to the variable.

- **bool**
- – Stores either value true or false.
- **char**
- – Typically a single octet (one byte). This is an integer type.
- **int**
- – The most natural size of integer for the machine.
- **float**
- – A single-precision floating point value.
- **double**
- – A double-precision floating point value.
- **void**
- – Represents the absence of type.
- **wchar_t**
- – A wide character type

Variable Definition in C++

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- A variable definition tells the compiler where and how much storage to create for the variable.
- A variable definition specifies a data type, and contains a list of one or more variables of that type.
- Syntax:

  type variable_list;

- Example:

```
int i, j, k;
char c, ch;
float f, salary;
double d;
```

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- Variables can be initialized (assigned an initial value) in their declaration.
- The initializer consists of an equal sign followed by a constant expression
- Syntax:

type variable_name = value;

- Example:
- extern int d = 3, f = 5; // declaration of d and f.
- int d = 3, f = 5; // definition and initializing d and f.
- byte z = 22; // definition and initializes z.
- char x = 'x'; // the variable x has the value 'x'.

Variable Scope in C++

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- A scope is a region of the program and broadly speaking there are three places, where variables can be declared −
- – Inside a function or a block which is called local variables,
- – In the definition of function parameters which is called formal

  parameters.
- – Outside of all functions which is called global variables

2.7 C++ Constants/Literals

- Constants refer to fixed values that the program may not alter and they are called literals.
- Constants can be of any of the basic data types and can be divided into Integer Numerals, Floating-Point Numerals, Characters, Strings and Boolean Values.
- Again, constants are treated just like regular variables except that their values cannot be modified after their definition

2.7.1 Integer Literals

- An integer literal can be a decimal, octal, or hexadecimal constant.
- A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal.
- An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.
- 212 // Legal
- 215u // Legal
- 0xFeeL // Legal
- 078 // Illegal: 8 is not an octal digit
- 032UU // Illegal: cannot repeat a suffix

2.7.2 Floating-point Literals

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part.
- Represent floating point literals either in decimal form or exponential form.
- 3.14159 // Legal
- 314159E-5L // Legal
- 510E // Illegal: incomplete exponent
- 210f // Illegal: no decimal or exponent
- .e55 // Illegal: missing integer or fraction

2.7.3 Boolean Literals

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- There are two Boolean literals and they are part of standard
- C++ keywords –
- A value of true representing true.
- A value of false representing false

2.7.4 Character Literals

- Character literals are enclosed in single quotes.
- Escape sequence Meaning
- \\ -- \ character
- \’ -- ' character
- \" -- " character
- \? -- ? character
- \a -- Alert or bell
- \b -- Backspace
- \f -- Form feed
- \n -- Newline
- \r -- Carriage return
- \t -- Horizontal tab
- \v -- Vertical tab
- \o -- o Octal number of one to three digits
- \x -- h . . . Hexadecimal number of one or more digits

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

2.8 Defining Constants

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

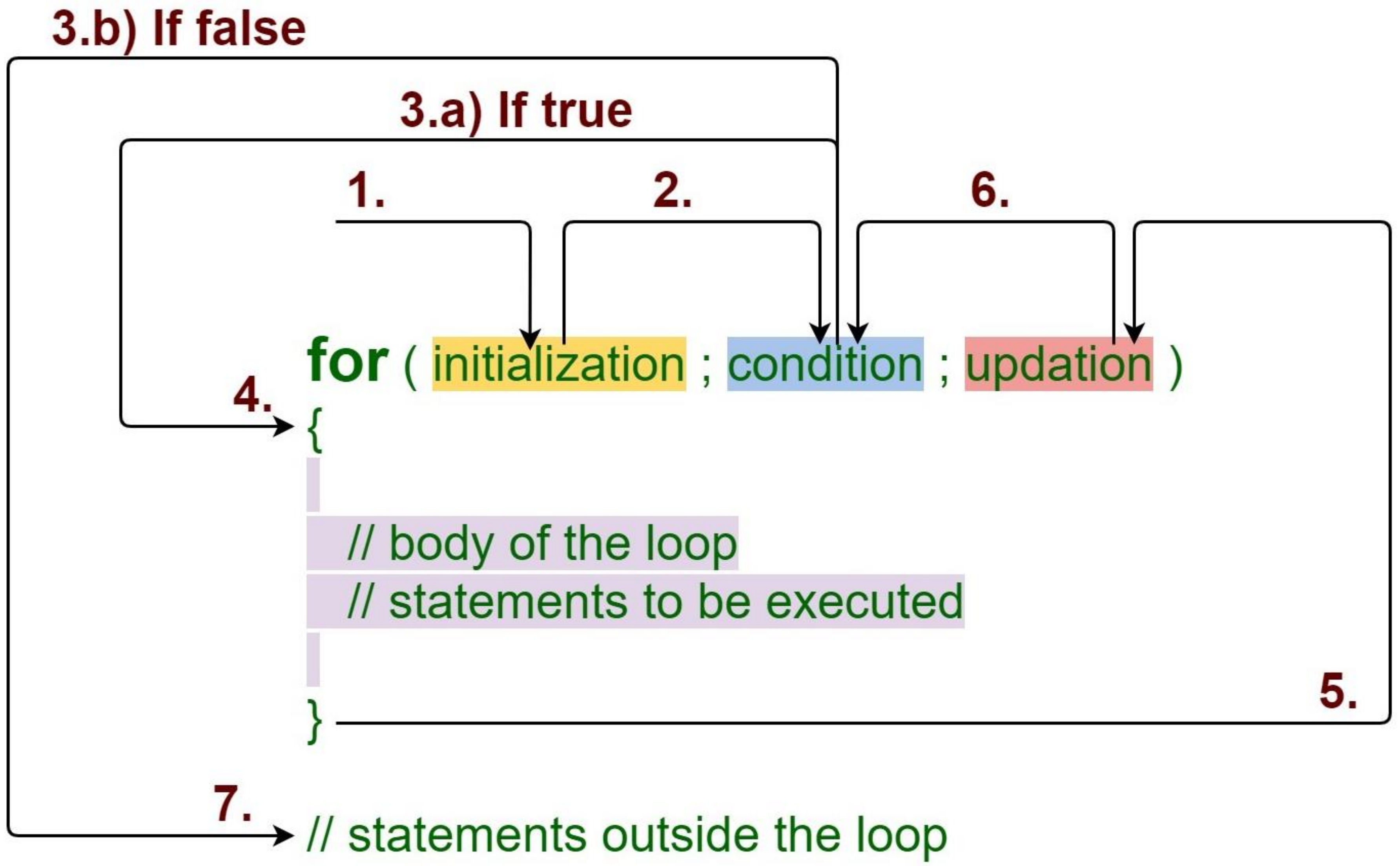
- Using #define preprocessor.
- – #define identifier value

- Using const keyword.
- – const type variable = value;

2.9 C++ Loop Types

- There may be a situation, when you need to execute a block of code several number of times.
- In general, statements are executed sequentially:
- The first statement in a function is executed first, followed by the second, and so on.
- Programming languages provide various control structures that allow for more complicated execution paths.
- There are two types of loops:
- Entry Controlled – While & for loop
- Exit Controlled – Do while Loop

For Loop


3.b) If false
3.a) If true
1.
2.
6.
for (initialization ; condition ; updation)
4.
{
// body of the loop
// statements to be executed
}
5.
7.
// statements outside the loop

Syntax

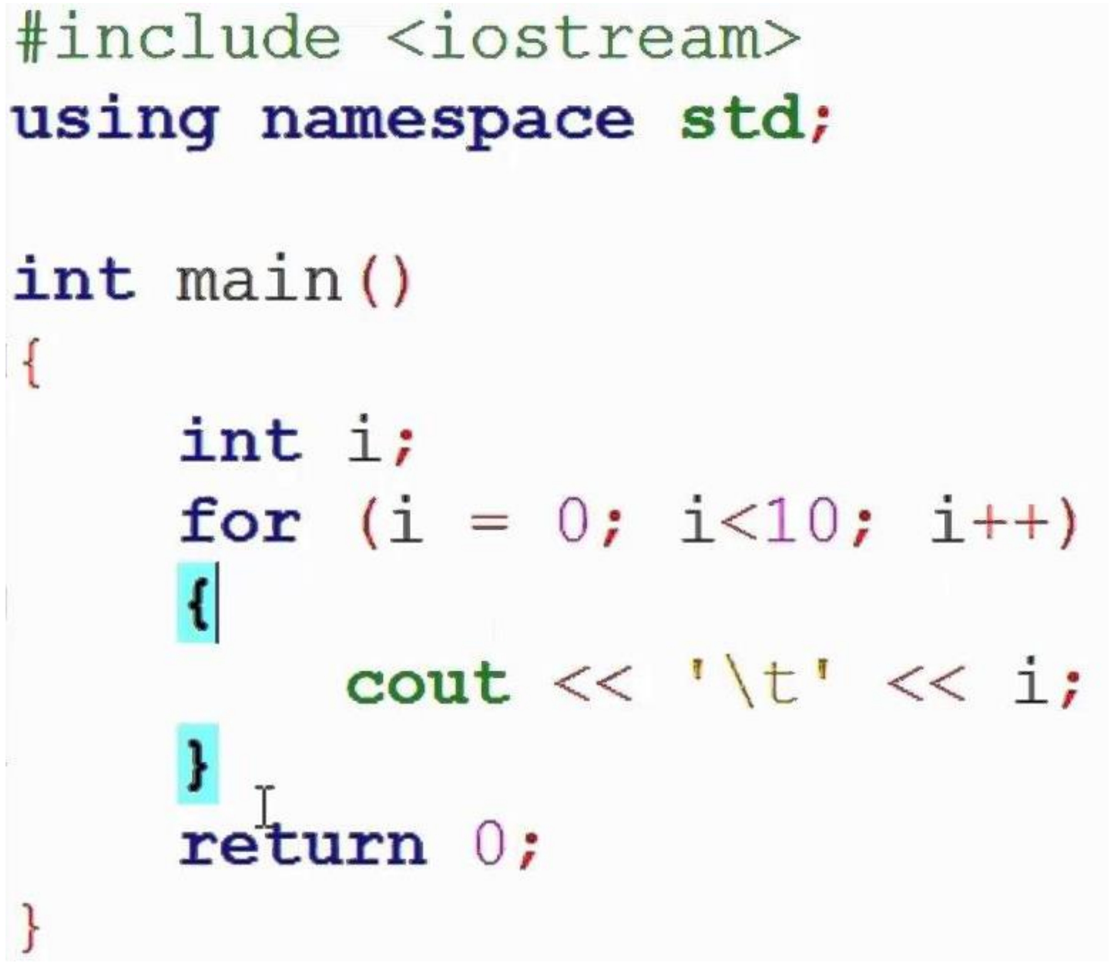
for (initialization ;   condition ;   increment)

{

// Body loop

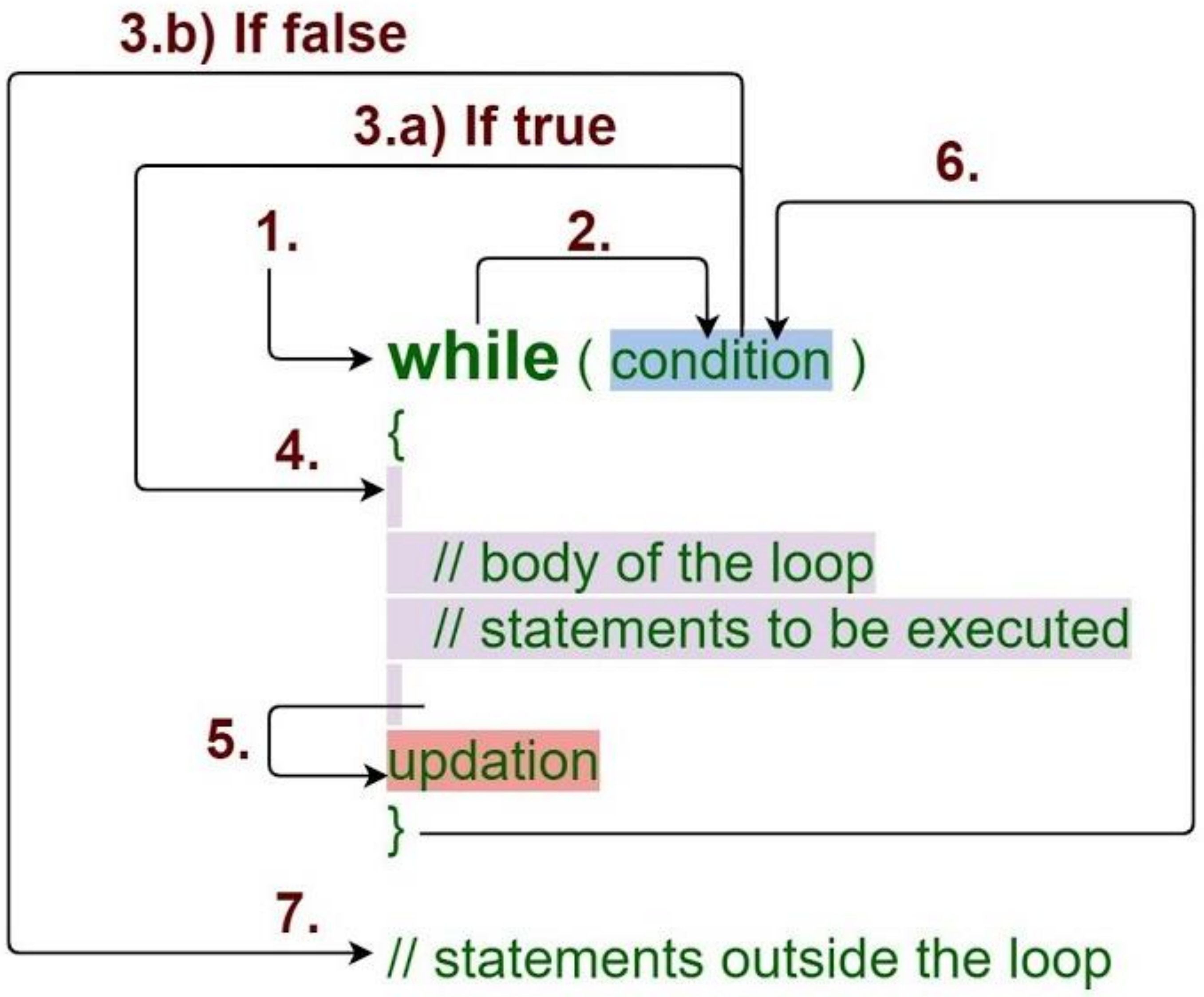
}

- Example: →



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

int main()
{
    int i;
    for (i = 0; i<10; i++)
    {
        cout << '\t' << i;
    }
    return 0;
}
```

While Loop

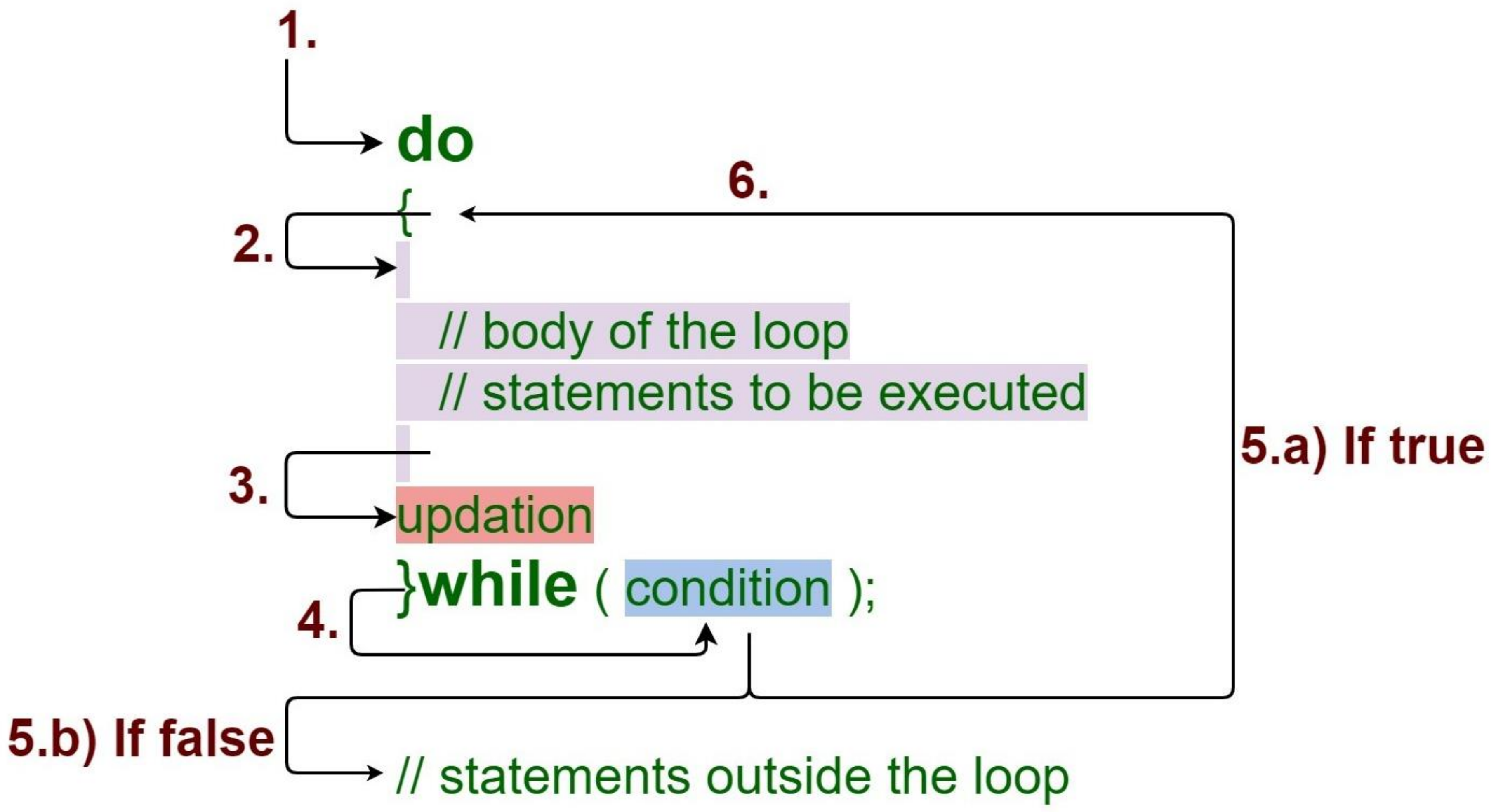


Syntax:

```
while ( condition )
{
// Statements
}
```

Do - While Loop

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade



Syntax

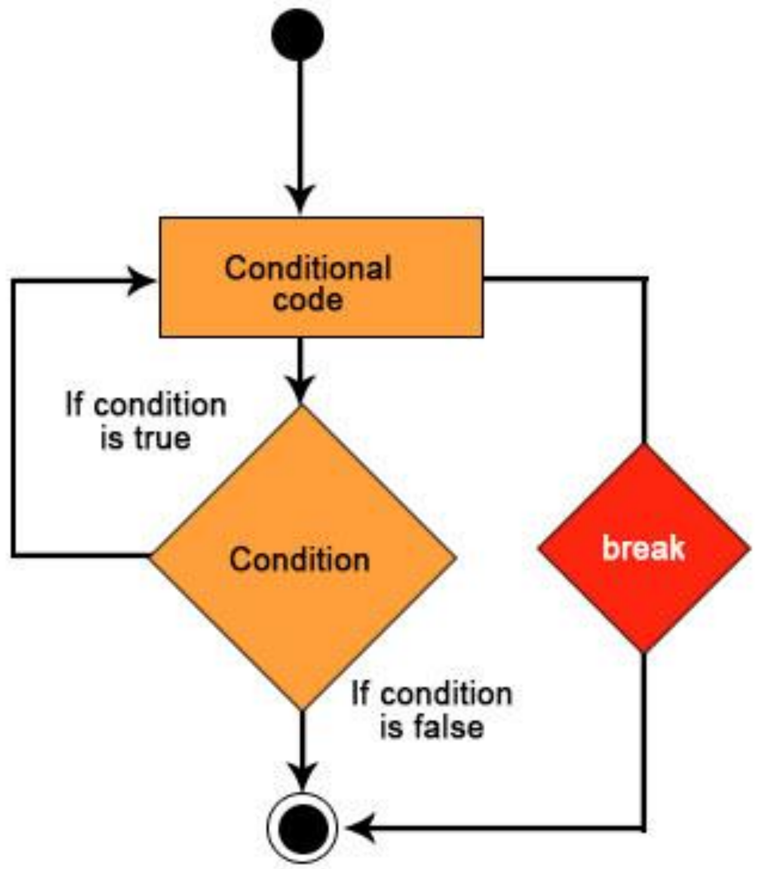
```
do {
//Body of the loop
}while (condition);
```

2.10 Control Statements

- 1. Break Statement
- 2. Continue Statement
- 3. Goto Statement

2.10.1 Break statement

- Break statement is used to terminate loop or switch statements.
- It is used to exit a loop early, breaking out of the enclosing curly braces.
- break;



2.10.2 Continue Statement

- Continue statement skips the remaining code block.
- This statement causes the loop to continue with the next iteration.
- It is used to suspend the execution of current loop iteration and transfer control to the loop for the next iteration.
- **Syntax:**

```
continue;
```

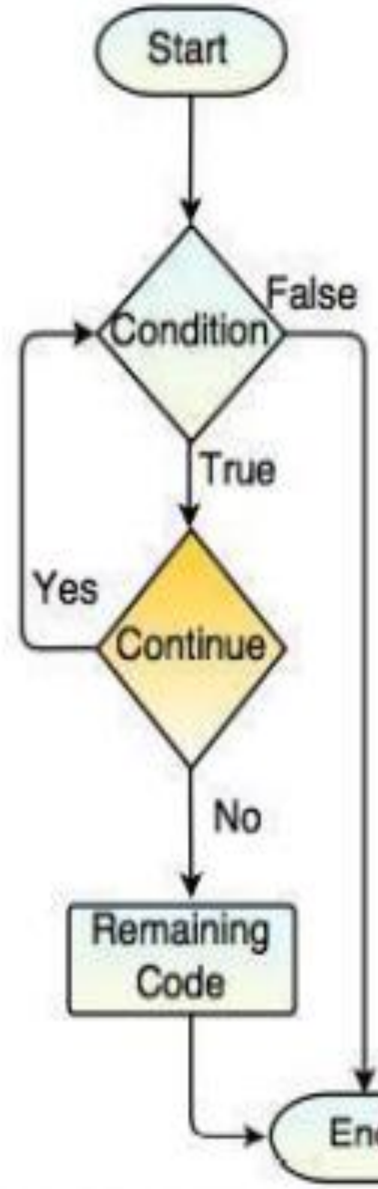


Fig. Flow Diagram of Continue Statement

- In For Loop, continue statement causes the conditional test and increment/ decrement statement of the loop gets executed.

- In While Loop, continue statement takes control to the condition statement.

- In Do-While Loop, continue statement takes control to the condition statement specified in the while loop.

2.10.3 Goto Statement

- Goto statement transfers the current execution of program to some other part.
- It provides an unconditional jump from goto to a labeled statement in the same function.
- It makes difficult to trace the control flow of program and should be avoided in order to make smoother program.
- Syntax:

```
goto label;
 ……
 label: Statement;
```

- It is used to exit from deeply nested looping statements.
- If we avoid the goto statement, it forces a number of additional tests to be performed.

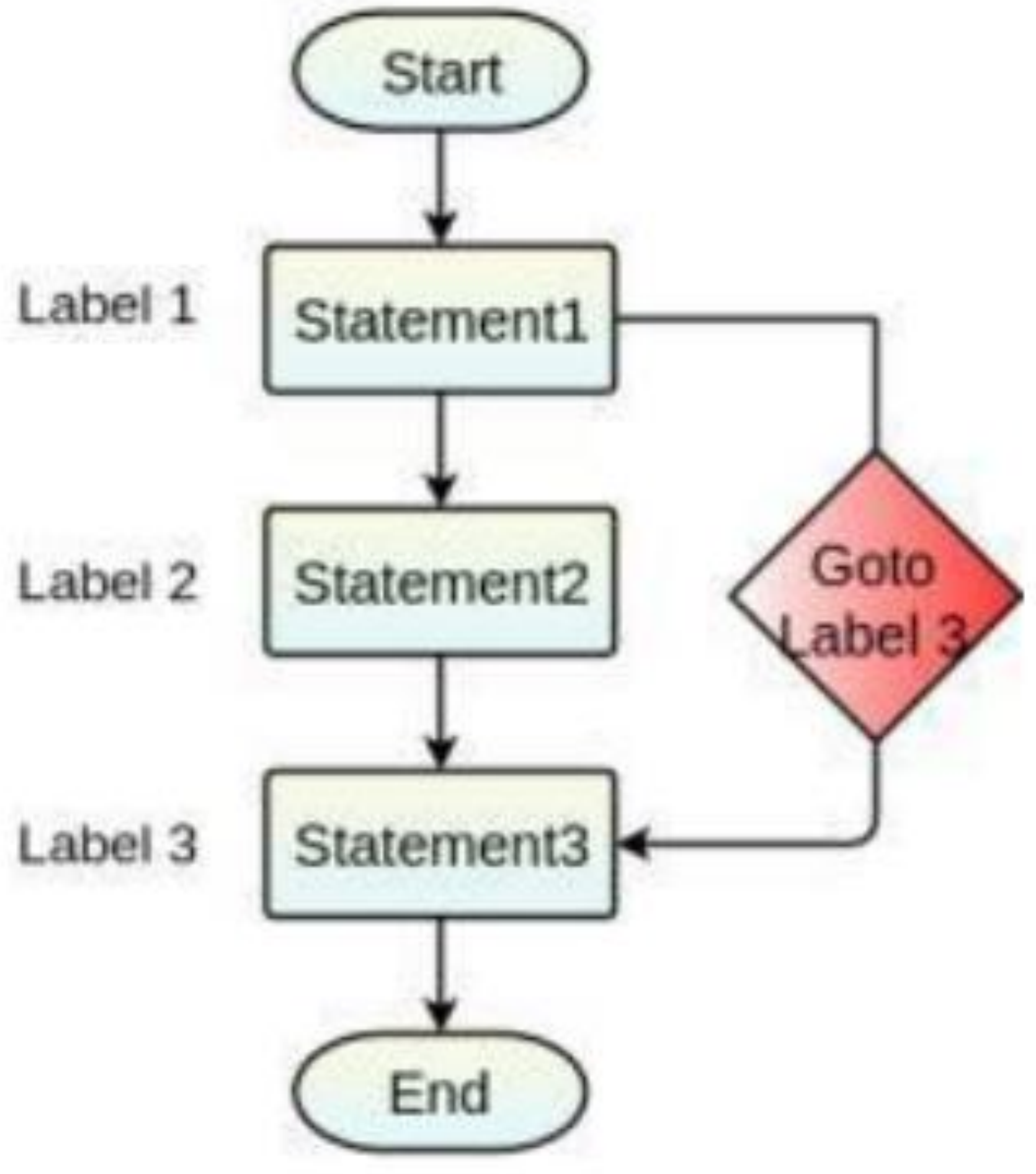
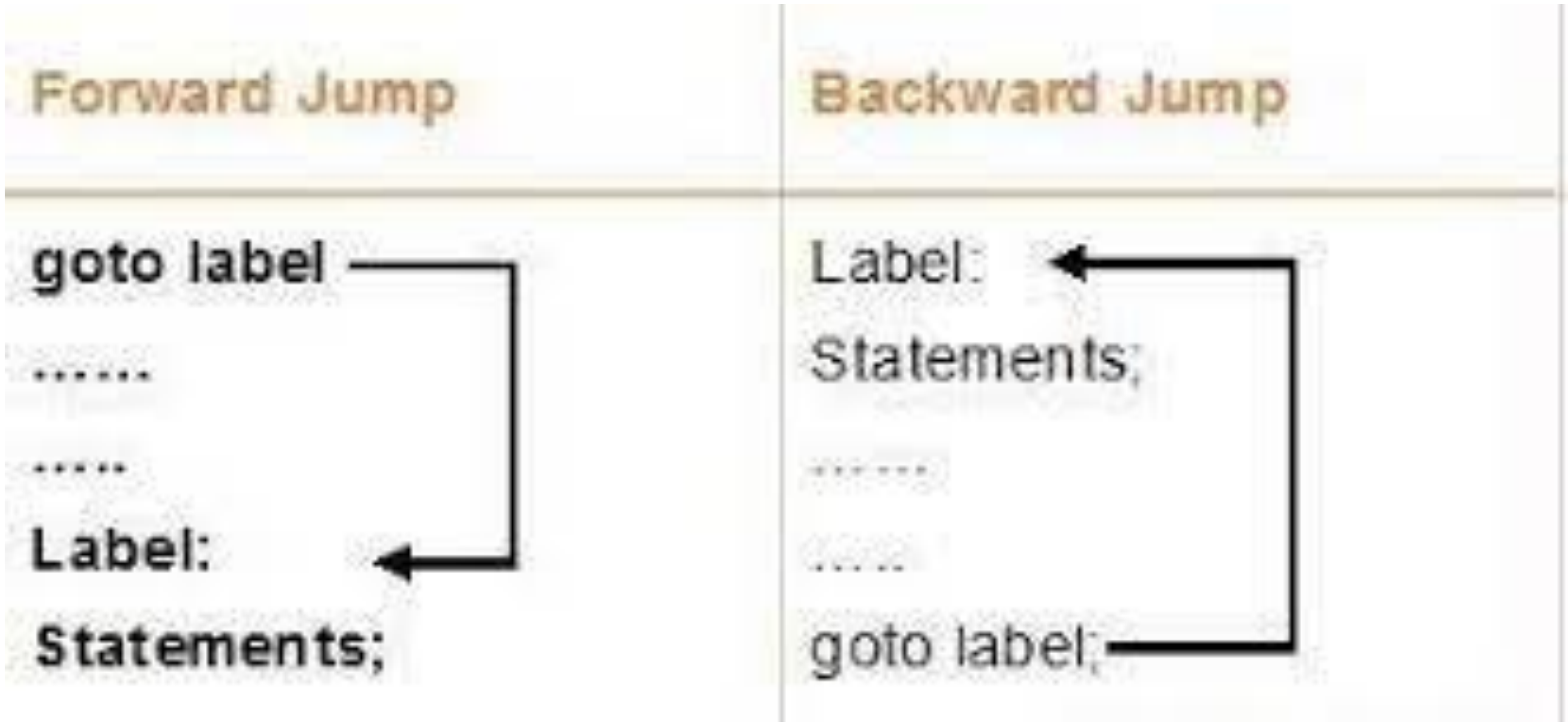


Fig. Flow Diagram of Goto Statement


Forward Jump
goto label
......
.....
Label:
Statements;
Backward Jump
Label:
Statements;
......
.....
goto label;

2.11 Decision Making Statements

- Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false

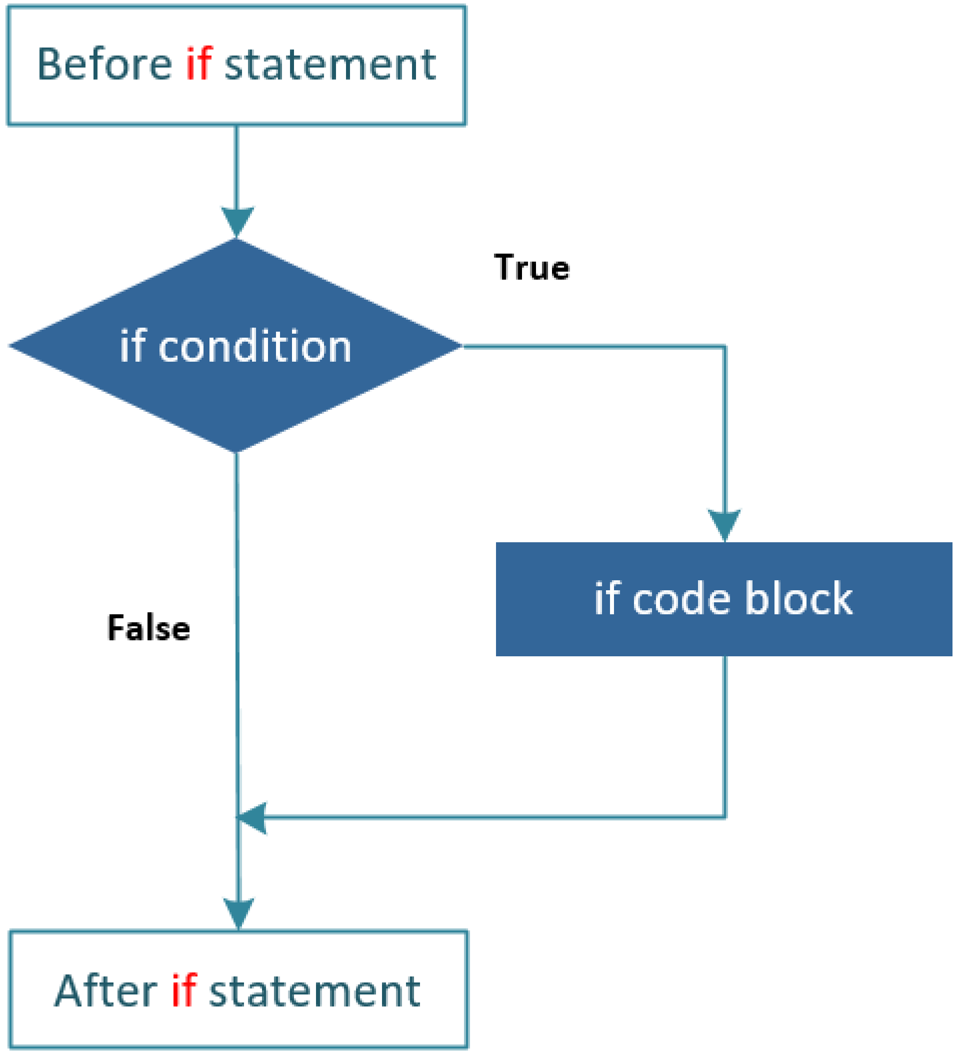
Sr.No	Statement & Description
1	if statementAn 'if' statement consists of a boolean expression followed by one or more statements.
2	if...else statementAn 'if' statement can be followed by an optional 'else' statement, which executes when the boolean expression is false.
3	switch statementA 'switch' statement allows a variable to be tested for equality against a list of values.
4	nested if statementsYou can use one 'if' or 'else if' statement inside another 'if' or 'else if' statement(s).
5	nested switch statementsYou can use one 'switch' statement inside another 'switch' statement(s).

2.11.1 If statement

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- An if statement consists of a boolean expression followed by one or more statements.

```
if(boolean_expression) {

    // statement(s) will execute if the boolean
expression is true
 }
//Outside the if statemen
```



Example:

```
#include <iostream.h>
int main ()
{
int a = 10;     // local variable declaration:
if( a < 20 )     // check the boolean condition
{
        cout << "a is less than 20 ” << endl;     // if condition is true
                                                   then print the following
}
cout << "value of a is : " << a << endl;
return 0;
}
```

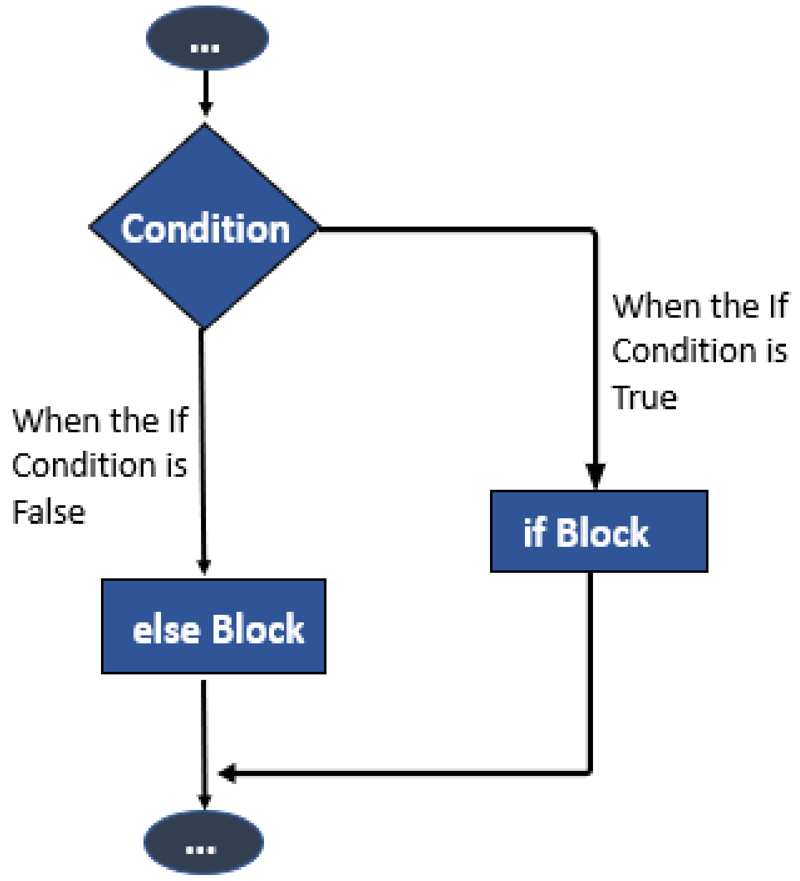
2.11.2 If…else

- An if statement can be followed by an optional else statement, which executes when the boolean expression is false.
- Syntax:

```
if(boolean_expression)
{
        // statement(s) will execute if the boolean expression is true
 }
else
{
      // statement(s) will execute if the boolean expression is false
}
```

Example:

```
#include <iostream.h>
int main ()
{
int a = 100; // local variable declaration:
if( a < 20 ) // check the boolean condition
{
           // if condition is true then print the following
          cout << "a is less than 20;" << endl;
} else
{
          // if condition is false then print the following
          cout << "a is not less than 20;" << endl;
}
cout << "value of a is : " << a << endl;
return 0;
}
```



2.11.3 if...else if...else Statement

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- An if statement can be followed by an optional else if...else statement, which is very usefull to test various conditions using single if...else if statement.
- When using if , else if , else statements there are few points to keep in mind.
- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of he remaining else if's or else's will be tested.

Syntax:

```
if(boolean_expression 1) {
// Executes when the boolean expression 1 is true
} else if( boolean_expression 2) {
// Executes when the boolean expression 2 is true
} else if( boolean_expression 3) {
// Executes when the boolean expression 3 is true
} else {
// executes when the none of the above condition is true.
}
```

Example:

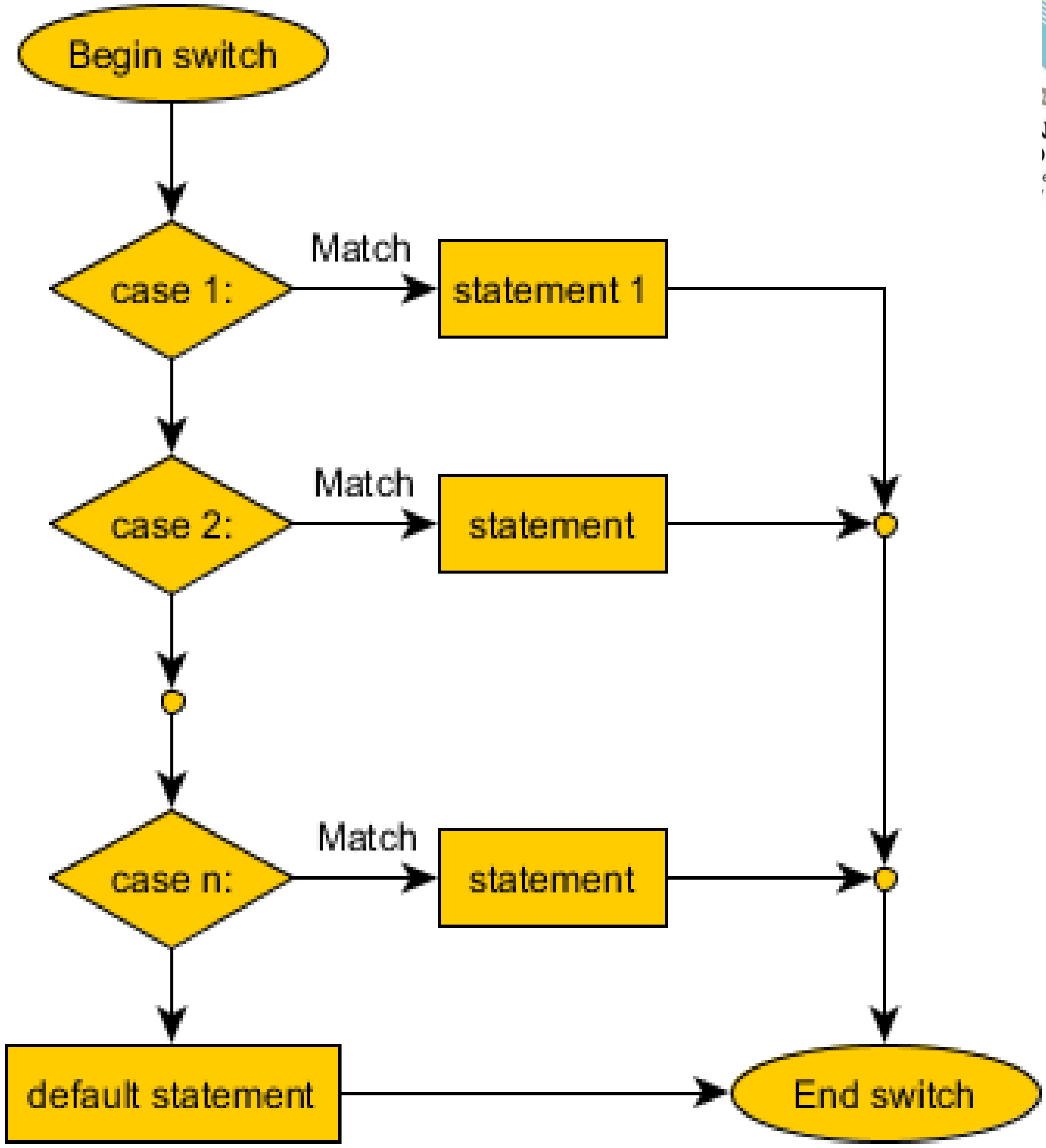
```
#include <iostream.h>
int main ()
{
int a = 100;              // local variable declaration:
if( a == 10 ) // check the boolean condition
{
            cout << "Value of a is 10" << endl;     // if condition is true then print the following
} else if( a == 20 ) {
            cout << "Value of a is 20" << endl;    // if else if condition is true
} else if( a == 30 ) {
            cout << "Value of a is 30" << endl;    // if else if condition is true
} else {
            cout << "Value of a is not matching" << endl;   // if none of the conditions is true
}
cout << "Exact value of a is : " << a << endl;
return 0;
}
```

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

2.11.4 Switch Stmnt

- Select one of many code blocks to be executed.
- **Syntax:**

```
switch(expression) {
    case x: // code block
            break;
    case y: // code block
            break;
    default: // code block
}
```



```
int day = 4;
switch (day) {
case 1:
    cout << "Monday";     break;
case 2:
cout << "Tuesday";        break;
case 3:
cout << "Wednesday";   break;
case 4:
cout << "Thursday"; break;
case 5:
cout << "Friday"; break;
case 6:
cout << "Saturday";  break;
case 7:
cout << "Sunday";  break;
}
// Outputs "Thursday" (day 4)
```

2.12 Scope resolution operator in C++

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- The **:: (scope resolution)** operator is used to get hidden names due to variable scopes so that you can still use them.

- The scope resolution operator can be used as both unary and binary.

- You can use the unary scope operator if a namespace scope or global scope name is hidden by a particular declaration of an equivalent name during a block or class.

Example 1:

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

```
#include <iostream.h>
int var = 0;
int main(void) {
int  var = 0;
::var = 1;        // set global   var to 1
var = 2;          // set local     var to 2
cout << ::var << ", " << var;
return 0;
}
```

Example 2:

```
#include <iostream.h>
class X {
public:
        static int count;
};

int X::count = 10;     // define static data member
int main () {
int X = 0;             // hides class type X
cout << X::count << endl;          // use static member of class X
}
```

2.13 C++ Modifier Types

- C++ allows the char, int, and double data types to have modifiers preceding them.
- A modifier is used to alter the meaning of the base type so that it more precisely fits the needs of various situations.
- signed
- unsigned
- long
- short

- The modifiers signed, unsigned, long, and short can be applied to integer base types. In addition, signed and unsigned can be applied to char, and long can be applied to double.

- The modifiers signed and unsigned can also be used as prefix to long or short modifiers.

- For example, unsigned long int.

Example:

```
#include <iostream.h>
/* This program shows the difference between
* signed and unsigned integers.
*/
int main() {
short int i;        // a signed short integer
short unsigned int j;   // an unsigned short integer
j = 50000;
i = j;
cout << i << " " << j;
return 0;
}
```

- **const**
- Objects of type const cannot be changed by your program during execution.
- **volatile**
- The modifier volatile tells the compiler that a variable's value may be changed in ways not explicitly specified by the program.
- **restrict**
- A pointer qualified by restrict is initially the only means by which the object it points to can be accessed. Only C99 adds a new type qualifier called restrict

2.14 Type Casting operators in C++

- A cast is a special operator that forces one data type to be converted into another.
- As an operator, a cast is unary and has the same precedence as any other unary operator
- Syntax:

(datatype) expression

or

expression (datatype)

Example:

```
#include <iostream.h>
main() {
double a = 21.09399;
float b = 10.20;
int c ;
c = (int) a;
cout << "Line 1 - Value of (int)a is :" << c << endl ;
c = (int) b;
cout << "Line 2 - Value of (int)b is :" << c << endl ;
return 0;
}
```

2.15 Dynamic Initialization

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- The process of initializing a variable at the moment it is declared at runtime is called dynamic initialization of the variable.
- Thus, during the dynamic initialization of a variable, a value is assigned to execution when it is declared.
- Dynamic initialization of object refers to initializing the objects at run time i.e. the initial value of an object is to be provided during run time.

- **Advantages:**
- It utilizes memory efficiently.
- Various initialization formats can be provided using overloaded constructors.
- It has the flexibility of using different formats of data at run time considering the situation.

Example:

```
#include<iostream>
int main()
{
int a;
cout<<“Enter Value of a”;
cin>>a;
int cube = a * a * a;
cout<<“Cube of “<< “ a:“<< cube;
return 0;
}
```

In the above example, the variable cube is initialized at run time using expression a * a * a at the time of its declaration.

2.16 Input & Output Statements in C++

- **<iostream.h>**
- This file defines the **cin, cout, cerr** and **clog** objects, which correspond to the standard input stream, the standard output stream, the un-buffered standard error stream and the buffered standard error stream, respectively.
- cin → Standard input stream
- cout →Standard output stream
- cerr → Standard output stream for errors
- clog → Standard output stream for logging

The Standard Input Stream (cin)

```
#include <iostream.h>
int main()
 {
char name[50];
cout << "Please enter your name: ";
cin >> name;
cout << "Your name is: " << name << endl;
}
```

- **<iomanip.h>**
- iomanip is a library that is used to manipulate the output of C++ program.

- **setfill**
- It is used to set fill character.

- **setprecision**
- It is used to set decimal precision.

- **setw**
- It is used to set field width.

Example 1:

```
#include <iostream.h>
#include <iomanip.h>
int main () {
    cout << setfill ('x') << setw (10);
    cout << 77 << endl;
 return 0;
}
```

Output:

```
xxxxxxxx77
```

Example 2:

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

```
#include <iostream>
#include <iomanip>
int main () {
double f =3.14159;
cout << setprecision(5) << f << '\n';
cout << setprecision(9) << f << '\n';
cout << fixed;
cout << setprecision(5) << f << '\n';
cout << setprecision(9) << f << '\n';
return 0;
}
```

Output:

```
3.1416
3.14159
3.14159
3.141590000
```

Example 3:

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

```
#include <iostream>

#include <iomanip>

int main () {

cout << setw(10);

cout << 77 << endl;

return 0;

}
```

<cmath> functions:

- Declares a set of functions to perform mathematical operations
- **1. pow()**
- Returns the result of the first argument raised to the power of the second argument.
- Syntax: pow(a, b) = a^b.
- Return type : float, double, long.

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- **2. sqrt()**
- Returns the square root of a number.
- Syntax: sqrt(x)
- Return type: float, double, long

- **3. cbrt()**
- Returns the cube root of a number.
- Syntax: cbrt(x)
- Return type: float, double, long, long double.
- Ex: cbrt(27) = 3
- cbrt(96) = 4.57886

- **4. round()**
- Returns the integral value that is nearest to the argument, with halfway cases rounded away from zero.
- Syntax: round(x)
- Return type: int, long
- Ex: round(158.268) = 158
- round(25.5) = 26

Armstrong Number

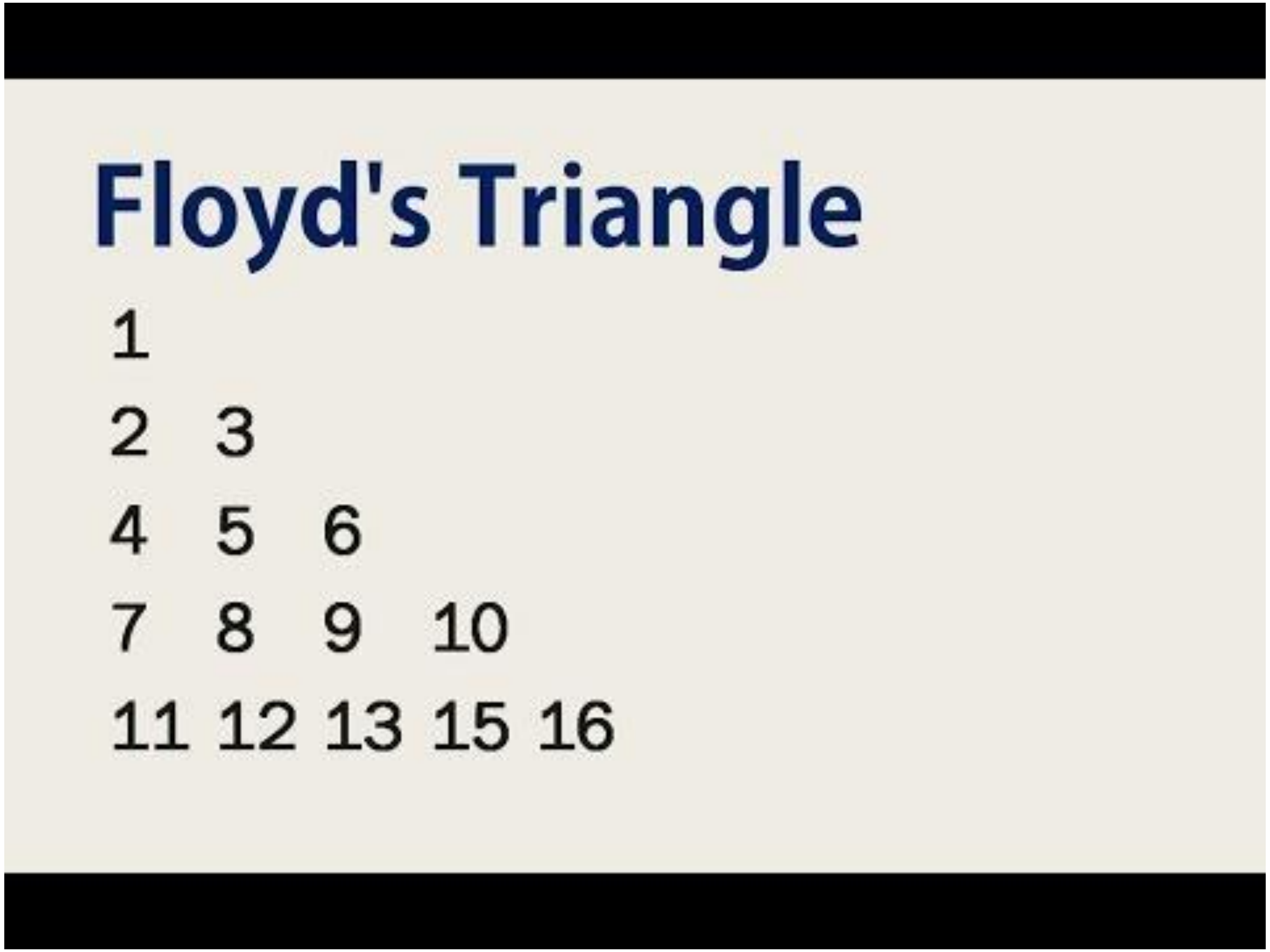
- A positive integer is called an Armstrong number (of order n) if

  $abcd... = a^n + b^n + c^n + d^n + ...$

- For example:
- 153 = 1*1*1 + 5*5*5 + 3*3*3
- 1634 = 1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4

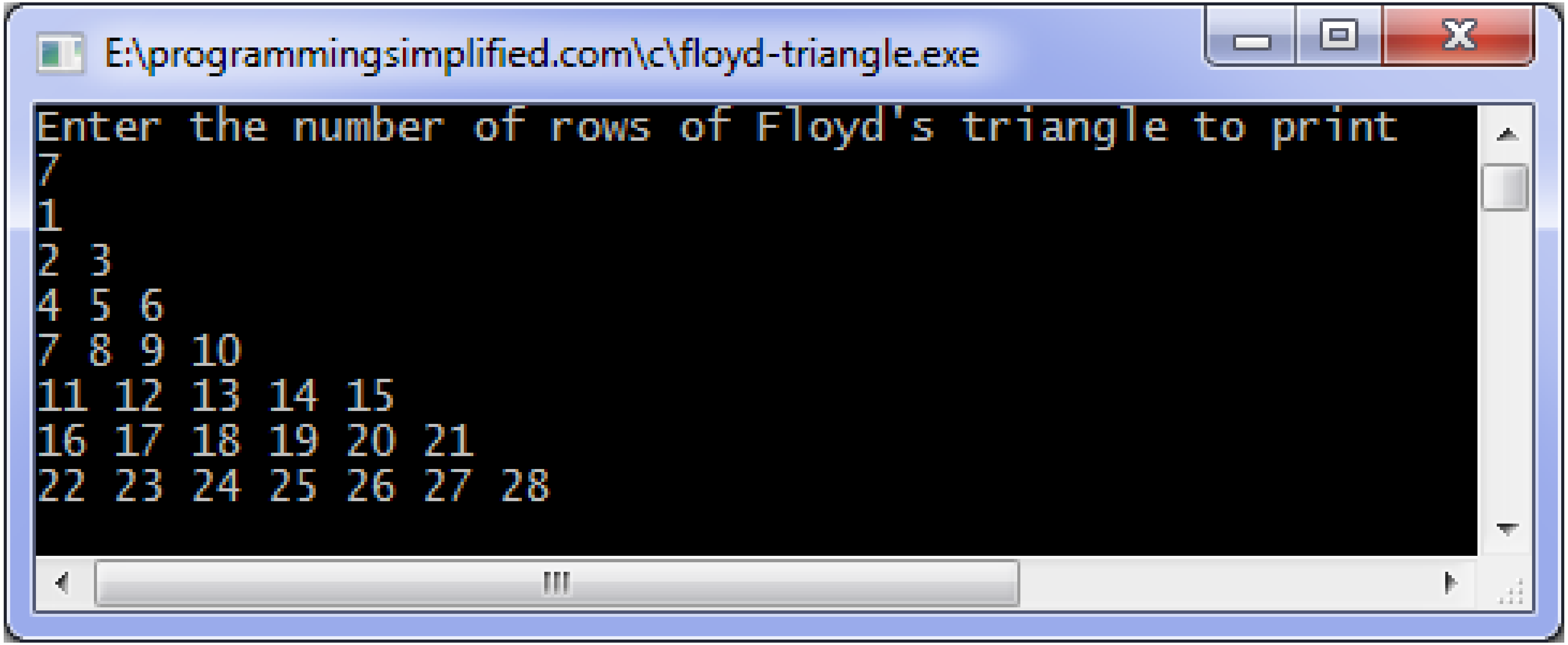
Dudeney number

- A number is a dudeney number, if the cube root of a number is equal to the sum of its digits.
- For example: Num = 19683
- Cube root of 19683 = 27
- Sum = 1+9+6+8+3 = 27
- There fore 19683 is a Dudeney Number.



Floyd's Triangle

```
1
2  3
4  5  6
7  8  9  10
11 12 13 15 16
```



```
Enter the number of rows of Floyd's triangle to print
7
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
22 23 24 25 26 27 28
```

```
n=1
for(i = 1; i <= num; i++)
   {
      for(int j = 1; j <= i; ++j)
      {
         cout << n << "      ";
         ++n;
      }
       cout << endl;
   }
```

Where,   num = Row number.

Happy Number

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

- A number is said to be Happy if and only if the sum of the square of its digits is equal to one.
- Replace the number by the sum of the squares of its digits, and repeat the process.
- At the end, if the number is equals to 1 then it is a Happy Number, or it loops endlessly in a cycle that does not include 1.
- if number is not a happy number then this process will end at 4

Example:

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

For example 1: 23

23 : 2^2 + 3^2 = 13

13 : 1^2 + 3^2 = 10

10 : 1^2 + 0^2 = 1

As we reached to 1, 23 is a Happy Number.

For example 2: 42

42 : 4^2 + 2^2 = 20

20 : 2^2 + 0^2 = 4

As we reached to 4, 42 is a UnHappy Numer.

Harshad Numbers

An n-harshad number is an integer number divisible by the sum of its digit.

For example: 153
1+5+3 = 9 & 153 % 9= 0 (i.e., 153 is completely divisible by 9)

15
1+5 = 6 & 15 % 6 = 3 (since 15 is not divisible by 6 it is not a Harshad Number)

Zeisel Numbers

Let us define a sequence as

$$\begin{cases} p_0 = 1 \\ p_n = a \cdot p_{n-1} + b, \end{cases}$$

where $a, b \in \mathbb{Z}$. If the numbers $p_1, p_2, \ldots, p_k$ are all distinct primes and $k \geq 3$, then their product is a *Zeisel number*.

p0=1, a=4, b=-1
p1 = a*p0 +b = 4*1-1 = 3
p2 = a*p1 +b = 4*3 -1= 11
p3 =a*p2 +b = 4*11-1= 43
3*11*43 = 1419 is a Zeisel number

Pascals Triangle

$n = 0$ (0th row)

$n = 1$ (1st row)

$n = 2$ (2nd row)

$n = 3$ (3rd row)

$n = 4$ (4th row)

$n = 5$ (5th row)

$${}_0C_0$$

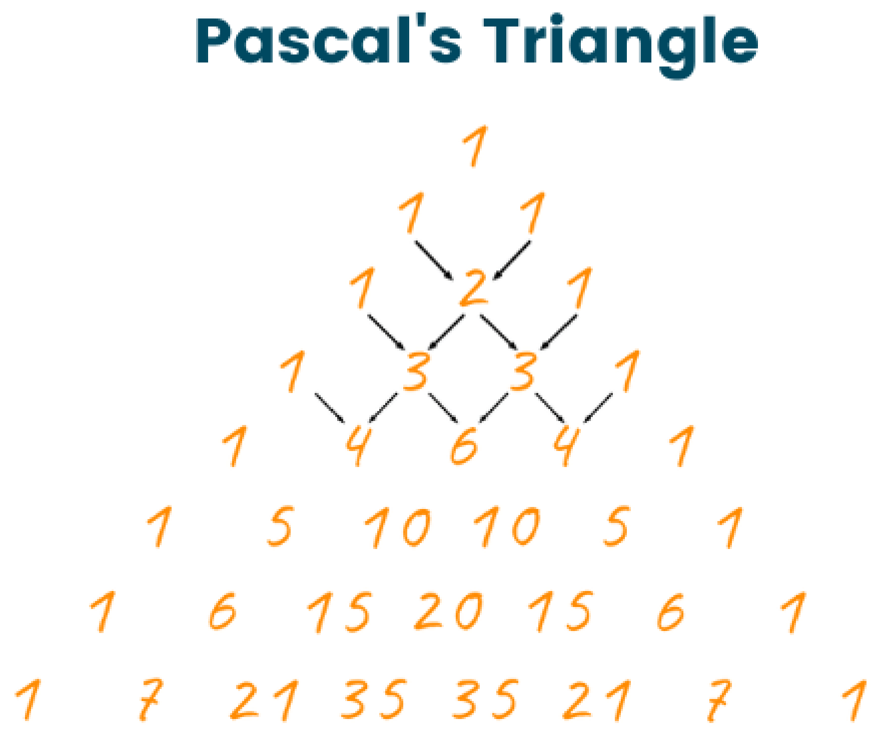
$${}_1C_0 \quad {}_1C_1$$

$${}_2C_0 \quad {}_2C_1 \quad {}_2C_2$$

$${}_3C_0 \quad {}_3C_1 \quad {}_3C_2 \quad {}_3C_3$$

$${}_4C_0 \quad {}_4C_1 \quad {}_4C_2 \quad {}_4C_3 \quad {}_4C_4$$

$${}_5C_0 \quad {}_5C_1 \quad {}_5C_2 \quad {}_5C_3 \quad {}_5C_4 \quad {}_5C_5$$



Ackermann function

$$A(m,n) = \begin{cases} n + 1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1,A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

where m and n are non-negative integers

Harshad Numbers

An n-harshad number is an integer number divisible by the sum of its digit.
For example: 153
1+5+3 = 9   & 153 % 9= 0 (i.e., 153 is completely divisible by 9)

15
1+5 = 6   & 15 % 6 = 3 (since 15 is not divisible by 6 it is not a Harshad Number)

Bernoulli’s Triangle

Each component of Bernoulli's triangle is the sum of two components of the previous row, except for the last number of each row, which is double the last number of the previous row.

$$
\begin{aligned}
B_{n,k} &= B_{n-1,k} + B_{n-1,k-1} && \text{if } k < n \\
B_{n,k} &= 2B_{n-1,k-1} && \text{if } k = n
\end{aligned}
$$

k / n	0	1	2	3	4	5
0	1					
1	1	2				
2	1	3	4			
3	1	4	7	8		
4	1	5	11	15	16	
5	1	6	16	26	31	32

Bell Triangle

```
1
1 2
2 3 5
5 7 10 15
15 20 27 37 52
```

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognised under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

To be Continued……