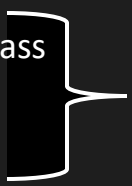
1. First c++ code
2. Escape sequence
3. Variable declare and initialization.
4. String print
5. User input
6. Arithmetic operator(+, -, *, /)
7. Output Formating
8. Area of Triangle.
9. Temperature conversion.
10. Assignment operator(+=, -+, *=, /+, %=)
11. Uniry operator(+, -, ++, --)
12. Bitwise operator(&, |, ^)
13. Bitwise operator(Left & Right shift)
14. Special operator(sizeof, comma)
15. Determine a Number is Positive or Negaive.
16. Even or Odd
17. Large number between two number.
18. Pass or Fail(33 marks easy)
19. Absolute value print.
20. Letter grade
21. Logical operator(&&, ||)
22. Vowel or Consonant
23. Large number between three number.
24. Leap Year.

25. Nested if
26. Large number between two number using Ternary operator.
27. Switch statement.
28. Vowel or Consonant using Switch
29. Simple for loop printing name five times.
30. Simple while loop
31. Simple do-while loop
32. Difference between while and do while loop.
33. break
34. continue
35. break and continue
36. Multiplication table
37. 1 + 2 + 3 + -------- + n
38. Basic pattern
39. Data type in C++
40. 1D array printing.
41. Take user input from an 1D array.
42. Finding sum, avg, max & min.
43. Types of Array
44. Printing a simple 2D array.
45. Taking user input from 2D array.
46. Pointer printing value and address.
47. Printing sum of two number using pointer.
48. Adding two number using function.

49. Add, sub, div, multi using function.
50. Return a value from a function.
51. Default parameter value.
52. Function without parameter.
53. Random number print.
54. Guessing game.
55. Function overloading.
56. Passing array through a function.
57. Factorial using recursion.
58. Call by value
59. Call by reference.
60. Taking string input using gets()
61. Scope resolution operator.
62. Understanding class & object
63. Class & object create example.
64. Using function inside the class.
65. Using parameterized function inside the class
66. Constructor example.
67. Default constructor with example.
68. String details with example
69. String library function.
70. String class(add str & finding size)
71. Creating separate files for class
72. Destructor



73. Selection operator
74. Consonant variable
75. Consonant object
76. Constructor initializer

Aygula(71-76) code deky na bujly Anisul Islam er video ta r ekbar deky nio.

77. Setter and getter.
78. This keyword
79. Intheritance theory and example.
80. Types of inheritance.
81. Function overriding.
82. Overloading vs Overriding difference.
83. Polymorphism theory and example with pointer.
84. UML class diagram.
85. Abstraction theory and example.
86. Friend class
87. Exception handling theory and example.
88. How to create and write into a file.
89. Storeing studens details in a file.
90. Read from a file

Objectives

1. What is C++ ?
2. What are the usages of C++?
3. What is the History of C++ ?
4. What are the Features of C++ ?

What is C++ ?

- C++ is known to be a very powerful computer programming language.
- C++ is a general purpose, case-sensitive , object oriented programming language.

Usage of C++

1. it is used to develop game engines, games, and desktop apps, art applications, music players etc.

2. C++ is being highly used to write device drivers and other software

History of C++

- **C++ programming language** was developed in 1980 by Bjarne Stroustrup at bell laboratories of AT&T.

- **Bjarne Stroustrup** is known as the **founder of C++ language.**

- C++ was derived from C, and is largely based on it.

Features of C++

1. Simple
2. Mid-level programming language
3. Rich Library
4. Memory Management
5. Fast Speed
6. Pointers
7. Recursion
8. Object Oriented
9. Compiler based

Click here to subscribe

C++ Bangla Tutorial

Translator program

আনিসুল ইসলাম
কম্পিউটার বিজ্ঞান ও প্রকৌশলী বিভাগ
anisul2010s@yahoo.co.uk

1

Dedicated to Suparna Bhattacharjee

Click here to subscribe

আজকের বিষয়বস্তু

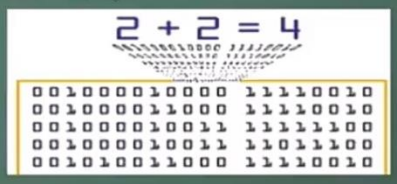
2

- ❑প্রোগ্রাম কি ? প্রোগ্রামিং ভাষা কি ?
- ❑অনুবাদক প্রোগ্রাম কি ? অনুবাদক প্রোগ্রাম এর প্রকারভেদ আলোচনা কর।
- ❑কোন কোন প্রোগ্রামিং ভাষাকে কম্পাইল বা ইন্টারপ্রেট করা হয় ?
- ❑কম্পাইলার কি? এটি কিভাবে কাজ করে ?
- ❑ইন্টারপ্রেটার কি? এটি কিভাবে কাজ করে ?
- ❑কম্পাইলার ও ইন্টারপ্রেটার এর পার্থক্য লিখ।

প্রশ্ন ১. প্রোগ্রাম কি ? প্রোগ্রামিং ভাষা কি ?

3

প্রোগ্রাম- কোন একটি সমস্যা সমাধানের জন্য কম্পিউটারের ভাষায় (0,1) লিখিত নির্দেশের সমষ্টিকে প্রোগ্রাম বলা হয়।



প্রোগ্রামিং ভাষা - প্রোগ্রাম রচনার জন্য বিভিন্ন শব্দ, বর্ণ, অঙ্ক, সঙ্কেত এবং এইগুলো বিন্যাসের নিয়ম মিলিয়ে তৈরি করা হয় প্রোগ্রামিং ভাষা।

যেমন - C, C++, Assembly language, Java ইত্যাদি ।

4

C++ programing language

যে প্রোগ্রাম সোর্স কোডকে অবজেক্ট কোডে রূপান্তরিত করে তাকে অনুবাদক প্রোগ্রাম বলে।

Translator program (অনুবাদক প্রোগ্রাম)

উচ্চতর ভাষায় লিখিত প্রোগ্রামকে সোর্স কোড (source code) বলে।

মেশিন ভাষার কোডকে অবজেক্ট কোড (Object code) বলে।

Click here to subscribe

5

প্রশ্ন ২. অনুবাদক প্রোগ্রাম কি ? অনুবাদক এর প্রকারভেদ আলোচনা কর।

অনুবাদক প্রোগ্রাম – যে প্রোগ্রাম সোর্স কোডকে অবজেক্ট কোডে রূপান্তরিত করে তাকে অনুবাদক প্রোগ্রাম বলে।

অনুবাদক প্রোগ্রাম

- ক) অ্যাসেম্বলার
- খ) কম্পাইলার
- গ) ইন্টারপ্রেটার

Click here to subscribe

জেনে রাখা ভাল

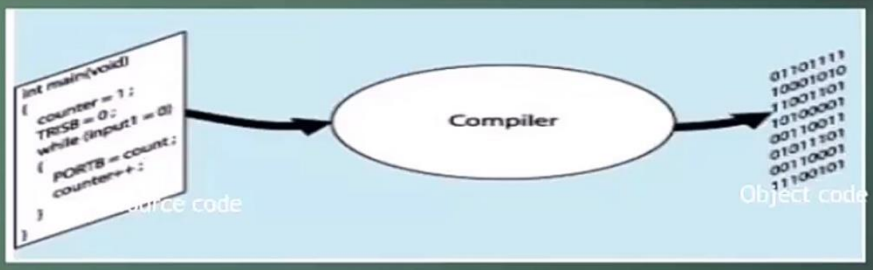
6

- যেসব প্রোগ্রামিং ভাষাকে কম্পাইল করা হয় -
  C, C++, Objective-C, C#, Pascal, COBOL, ADA, Visual Basic, Smalltalk, Scheme ইত্যাদি ।

- যেসব প্রোগ্রামিং ভাষাকে ইন্টারপ্রেট করা হয় -
  BASIC , php, Python, Perl, Ruby, Javascript ইত্যাদি ।

কম্পাইলার

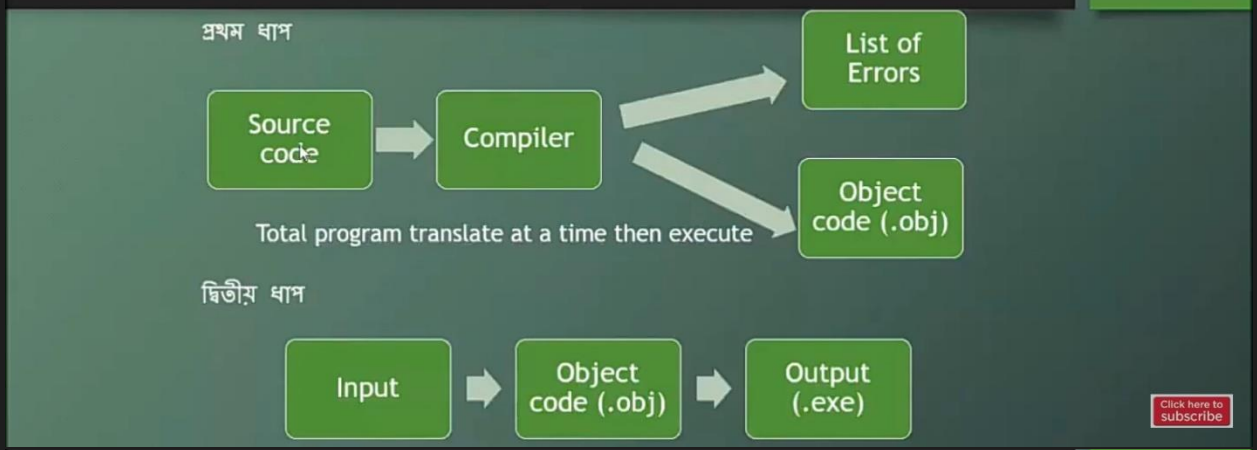
7



যে অনুবাদক প্রোগ্রাম সোর্স কোডকে অবজেক্ট কোডেরূপান্তরিত করে তাকে কম্পাইলার বলে।

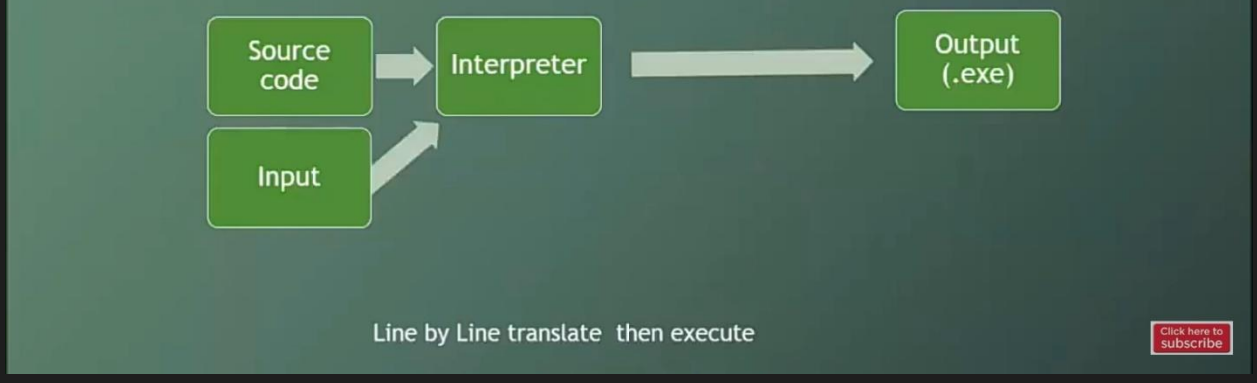
কম্পাইলারের কাজের ধাপসমূহ

8



ইন্টারপ্রেটারের কাজের ধাপ

9



কম্পাইলার ও ইন্টারপ্রেটার এর পার্থক্য লিখ।

11

কম্পাইলার	ইন্টারপ্রেটার
১. সম্পূর্ণ প্রোগ্রামটিকে একসাথে অনুবাদ করে।	১. এক লাইন এক লাইন করে অনুবাদ করে।
২. প্রোগ্রামের সব ভুল একসাথে প্রদর্শন করে।	২. এক লাইন করে ভুল প্রদর্শন করে।
৩. কম্পাইলার দ্রুত কাজ করে।	৩. ইন্টারপ্রেটার ধীরে কাজ করে।
৪. প্রোগ্রাম একবার কম্পাইল করার পর পরবর্তীতে আর কম্পাইল করার প্রয়োজন হয়না।	৪. প্রতিটি কাজের পূর্বে অনুবাদ করার প্রয়োজন হয়।
৫. বড় ধরণের কম্পিউটারে বেশি ব্যাবহার হয়।	৫. মাইক্রো কম্পিউটারে বেশি ব্যাবহার হয়।

প্রথম সি++ প্রোগ্রাম এবং তার বিভিন্ন অংশ

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

int main()
{
        cout << "Hello world!" ;
        return 0 ;
}
```

iostream = input output stream একটি হেডার ফাইল ।

#include এর সাহায্য এটি প্রোগ্রামে সংযুক্ত করা হয় ।

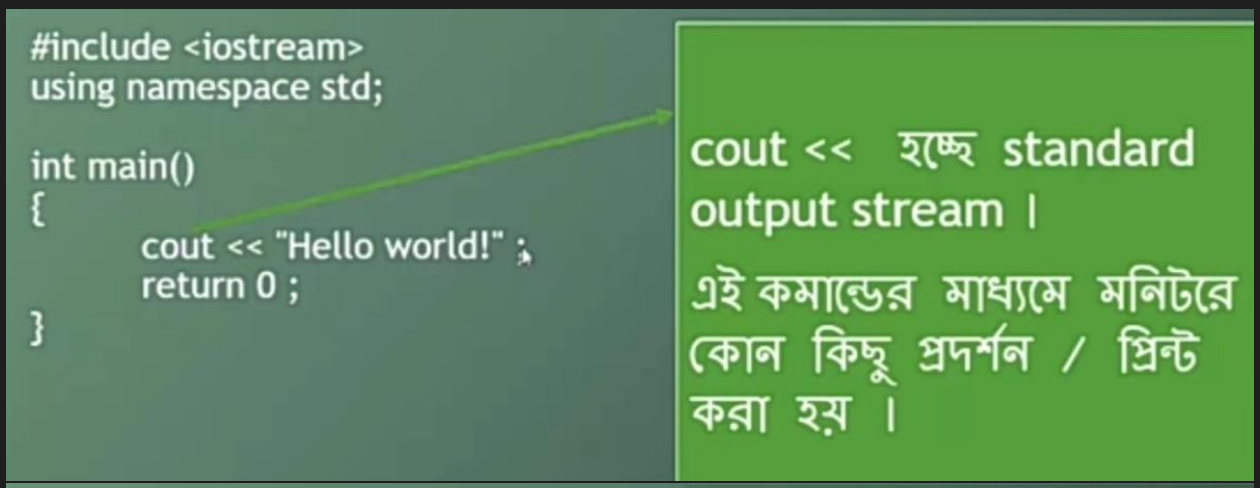
```
#include <iostream>
using namespace std;

int main()
{
        cout << "Hello world!" ;
        return 0 ;
}
```

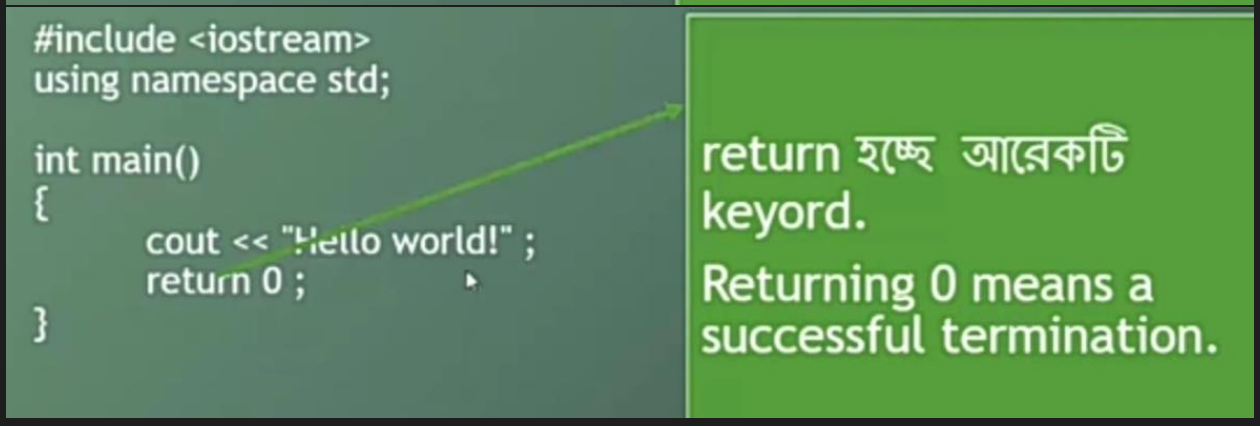
প্রোগ্রামের ভেতরে cout<< নামে একটি কমান্ড রয়েছে।

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

int main()
{
        cout << "Hello world!" ;
        return 0 ;
}
```

int = integer একটি keyword.

কম্পাইল এবং নির্বাহের সময় C/C++ প্রোগ্রামে main() ফাংশন থেকে শুরু হয়। তাই প্রোগ্রামে এ ফাংশন অবশ্যই লিখতে হয়।


#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" ;
return 0 ;
}
cout << হচ্ছে standard output stream ।
এই কমান্ডের মাধ্যমে মনিটরে কোন কিছু প্রদর্শন / প্রিন্ট করা হয় ।


#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" ;
return 0 ;
}
return হচ্ছে আরেকটি keyord.
Returning 0 means a successful termination.

1. First c++ code

```cpp
#include<iostream>
using namespace std;
int main(){

    cout << "Golam kibria";

    return 0;
}
```

```cpp
#include<iostream>
#include<conio.h>
using namespace std;
int main(){

    cout << "Golam kibria";

    getch();//output screec clear asby
}
```

```cpp
#include<iostream>
using namespace std;
int main(){

    cout << "Golam " << "kibria";

    return 0;
}
```

Output: Golam kibria

Escape sequence / backslash character

- সি++ প্রোগ্রামে কিছু Backslash Character ব্যাবহার করা হয় যা আউটপুট ফাংশন হিসাবে কাজ করে।
- সাধারণত Backslash (\ ) দেয়ার পর একটি ক্যারেক্টার ব্যাবহার করা হয়।

Character Escape Sequence	Meaning
\a	(Alert) Bell
\b	BackSpace
\f	Form Feed
\n	NewLine
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\0	Null Character
\'	Single Quote
\"	Double Quote
\\	Backslash
\?	Questionmark

2. Escape sequence

```
#include<iostream>
using namespace std;
int main(){

    cout << "\"Golam kibria\"\n";
    cout << "I live in Dhaka" << endl;
    cout << "017 9003 7447\n";
    cout << "Golam kibria" <<endl << "I live in
Dhaka" << endl << "017 9003 7447";
    return 0;
}//endl & new line er kaj same.
```

Output:

```
"Golam kibria"
I live in Dhaka
017 9003 7447
Golam kibria
I live in Dhaka
017 9003 7447
```

১. কমেন্ট কি ? প্রোগ্রামে কমেন্ট ব্যাবহার এর উদ্দেশ্য কি?
২. কমেন্ট এর প্রকারভেদ ও তাদের ব্যাবহার ।

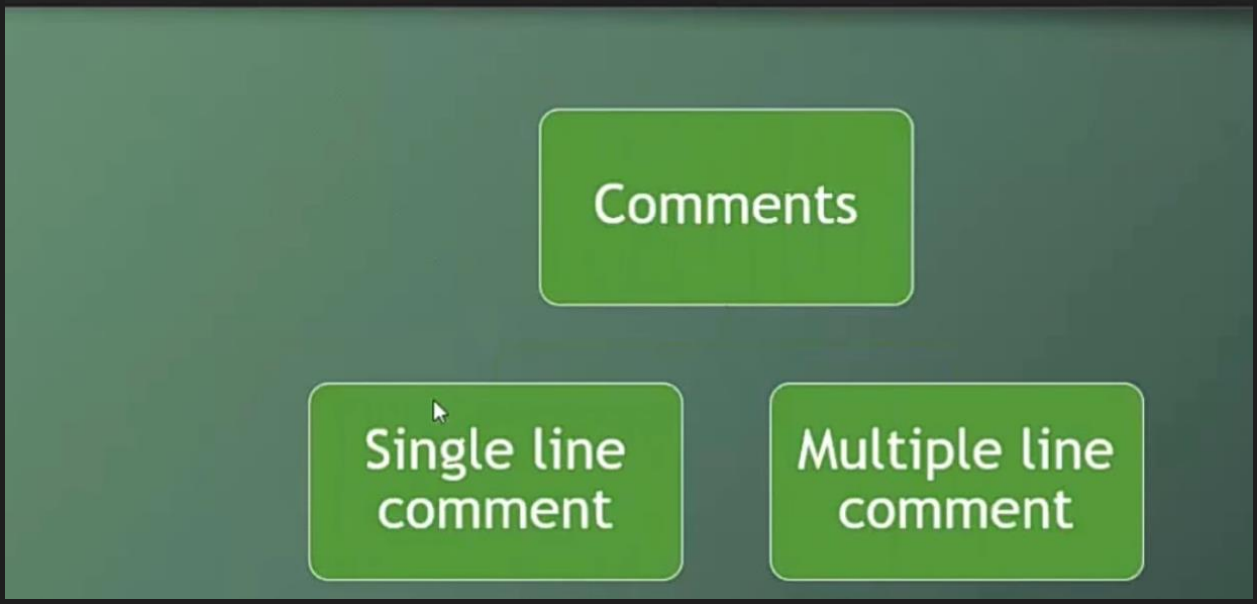
১. কমেন্ট কি ? প্রোগ্রামে কমেন্ট ব্যাবহার এর উদ্দেশ্য কি?

- কমেন্ট হচ্ছে সোর্স কোড লিখার সময় নিজের বা অন্যজনের জন্য একটি নোট তৈরি করে রাখা ।

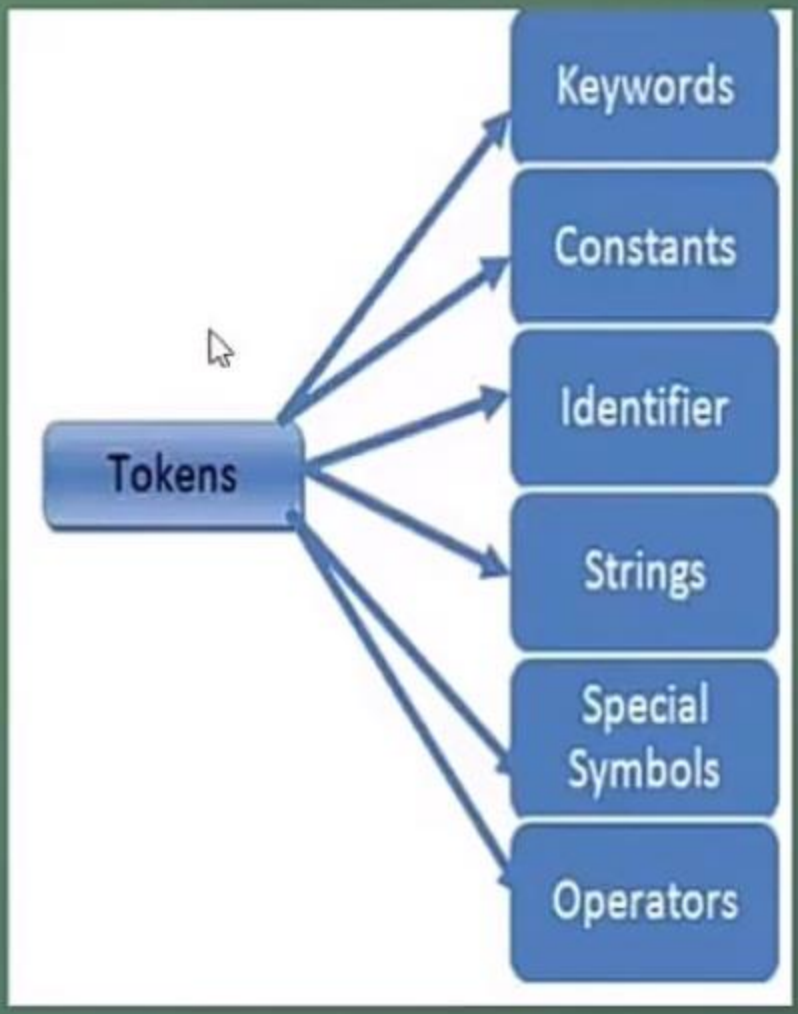
- প্রোগ্রামে কমেন্ট ব্যাবহার এর উদ্দেশ্য

❖প্রোগ্রামের উদ্দেশ্য সহজে বুঝানোর জন্য।
❖প্রোগ্রাম সম্পর্কে বর্ণনা লিখে রাখার জন্য ব্যাবহার হয়।
❖প্রোগ্রামের কোন লাইন কি বুঝায় তা বুঝানোর জন্য।

কমেন্ট এর প্রকারভেদ



সি++ এর টোকেন


Tokens
Keywords
Constants
Identifier
Strings
Special Symbols
Operators

কীওয়ার্ড (Keyword)

- কীওয়ার্ড : এমন কিছু Reserved Word বা সংরক্ষিত শব্দ যা একটি নির্দিষ্ট অর্থ বহন করে এবং প্রোগ্রামে একটি নির্দিষ্ট কার্যসম্পাদন করে।

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

কীওয়ার্ড (Keyword)

কীওয়ার্ড ব্যাবহার এর কিছু নিয়ম :

- কীওয়ার্ড কখনও variable বা চলকের নাম হিসাবে ব্যাবহার হয় না।
- কীওয়ার্ডসমূহের প্রতিটি বর্ণ ছোট হাতের হয়।

  যেমন : Int (Invalid keyword)

  int (Valid keyword)

- কখনও যদি দুইটি কীওয়ার্ড একত্রে ব্যাবহার হয় তবে তাদের মাঝে ফাঁকা স্থান থাকে।

  যেমন : longdouble (Invalid)

  long double (Valid)

- কীওয়ার্ডসমূহের নাম একটি একক শব্দ, অর্থাৎ মাঝে ফাঁকা কোন স্থান থাকেনা।

  যেমন : i n t (Invalid)

Variable/ভেরিয়েবল/চলক

5

প্রশ্ন : ভেরিয়েবল বা চলক কি ?

ভেরিয়েবল বা চলক :

প্রোগ্রামে কোন একটি নাম দিয়ে যদি তার অধীনে ডেটা রাখা হয়, তবে ঐ নামকেই চলক বলে।

Variable declaration syntax

---

data_type variable_name ;

int num;

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

int main()
{
    int num1,num2;

    num1=10;
    num2=20;

    int sum = num1 + num2;
    cout<<"Sum is : "<<sum;

    return 0;

}
```

ভেরিয়েবল লেখার নিয়মগুলো

1. ভেরিয়েবলের নামের মধ্যে বর্ণ (A...Z, a...z), অঙ্ক (0,1,.....,9), আন্ডারস্কোর (_), ডলারচিহ্ন($) ব্যাবহার করা যায়।

Variable name
role_number
“Anis”
Id1
anis@
subject102
#anis
$anis

Valid

---

role_number
Id1
Subject102
$anis

Invalid

“Anis”
anis@
#anis

ভেরিয়েবল লেখার নিয়মগুলো

২. ভেরিয়েবলের নাম ডিজিট বা অঙ্ক দিয়ে শুরু হতে পারেনা।

Variable name
1number
number1
_number1
a2b
96times

Valid

Number1
_number1
a2b

Invalid

1number
96times

৩. কোন কিওয়ার্ড ,ফাংশন ভেরিয়েবলের নাম হিসাবে ব্যাবহার করা যায় না।

Variable name
float
Float
main
MAIN
for

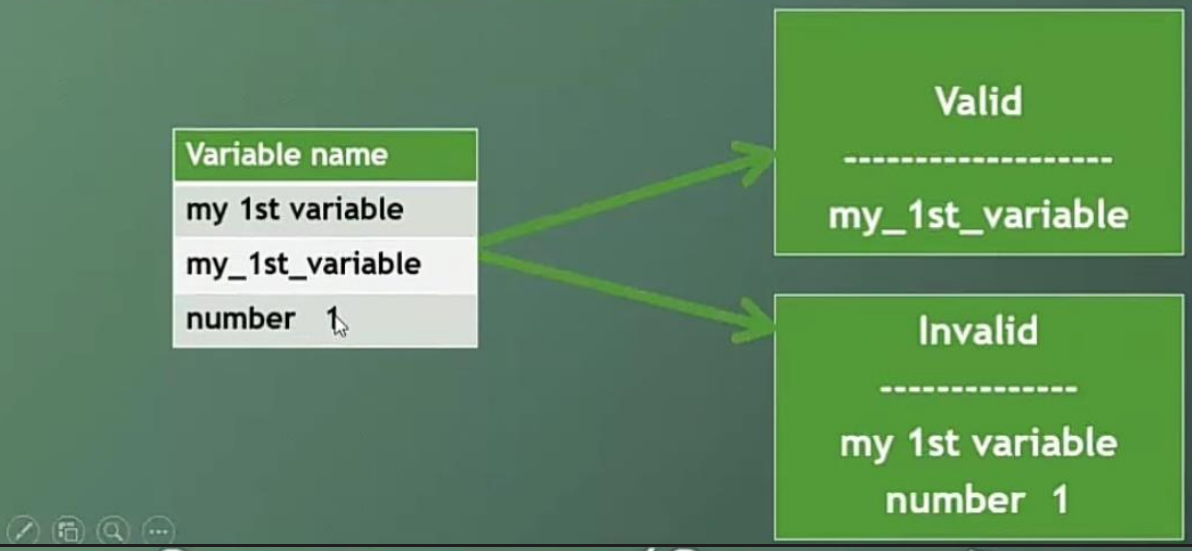
Valid

Float
MAIN

Invalid

Float
Main
for

৪. ভেরিয়েবলের নামের মধ্যে কোন ফাঁকা স্হান থাকতে পারেনা।



- ৫. ভেরিয়েবলের নামকরণে সর্বাধিক ৩১ ক্যারেক্টার ব্যাবহার করা যায়। তবে ৮ ক্যারেক্টার ব্যাবহার করাই শ্রেয়।

- ভেরিয়েবল / ফাংশন / অ্যারে লিখার নিয়মসমূহ -

১. ভেরিয়েবলের নামের মধ্যে বর্ণ (A...Z, a...z), অঙ্ক (0,1,.....,9), আন্ডারস্কোর (_), ডলারচিহ্ন($) ব্যাবহার করা যায়।

২. ভেরিয়েবলের নাম ডিজিট বা অঙ্ক দিয়ে শুরু হতে পারেনা।

৩. কোন কিওয়ার্ড ,ফাংশন ভেরিয়েবলের নাম হিসাবে ব্যাবহার করা যায় না।

৪. ভেরিয়েবলের নামের মধ্যে কোন ফাঁকা স্হান থাকতে পারেনা।

৫. ভেরিয়েবলের নামকরণে সর্বাধিক ৩১ ক্যারেক্টার ব্যাবহার করা যায়। তবে ৮ ক্যারেক্টার ব্যাবহার করাই শ্রেয়।

3. Variable declare and initialization.

```cpp
#include<iostream>
#include <iomanip>
using namespace std;
int main(){
    //akoi saty variable declare & assign koraky bola hoy dynamic initialization.
    int num1 = 10, num2 = 20;
    double d = 2.342345234;
    char c = 'K';

    cout << "Num1 is: " << num1 << endl;
    cout << "Num2 is: " << num2 << endl;
    cout << "Double number: " << setprecision(3) << d << endl;
    cout << "Character is: " << c << endl;

    return 0;
}
```

```
Num1 is: 10
Num2 is: 20
Double number: 2.34
Character is: K
```

4. String print

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    //string ta j ses hoyechy tai string terminator
er jonno akta null character use korty hoy.
    char name[13] = "Golam kibria";
    //string name = "Golam kibria";
    cout << "My name is " << name;
    return 0;
}
```

Output:

```
My name is Golam kibria
```

5. User input

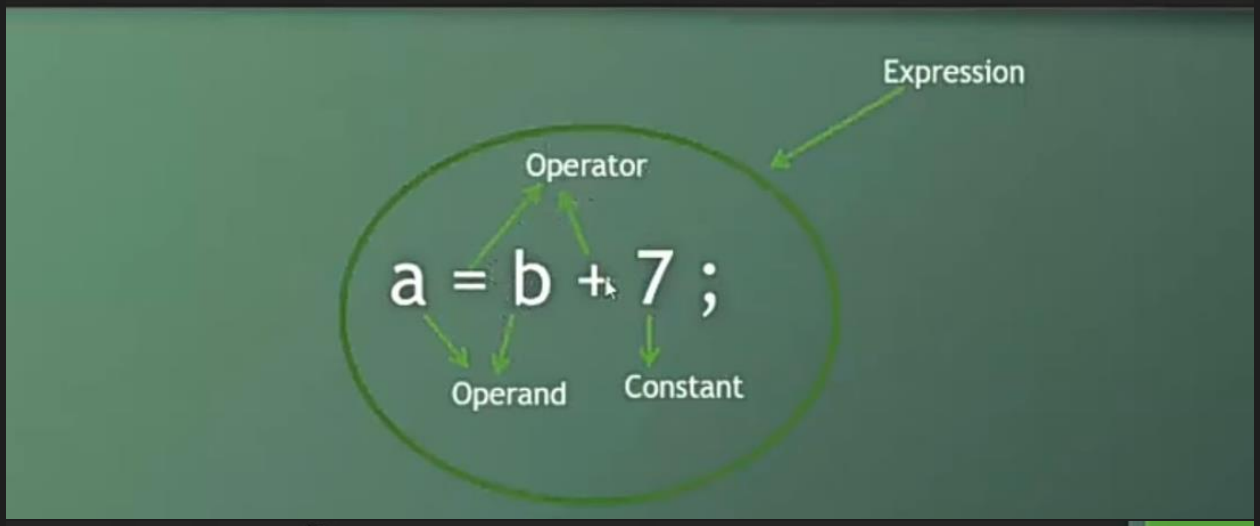
```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int num;
    char name[20];
    cout << "Enter an integer number: ";
    cin >> num;
    cout << "Enter your name: ";
    cin >> name;
    cout << "Enter number is: " << num << endl;
    cout << "Welcome " << name << endl;
}
Enter an integer number: 12
Enter your name: Golam kibria
Enter number is: 12
Welcome Golam
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int num1, num2, sum;
    cout << "Enter two number : ";
    cin >> num1 >> num2;
    sum = num1 + num2;
    cout << "The sum is : " << sum;
}
```

Enter two number : 12 23

The sum is : 35

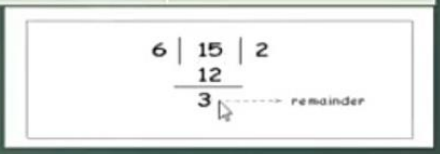
Example of Operator, Operand & Expression



২. অপারেটর এর প্রকারভেদ।

Arithmetic operator

অপারেটর	কাজ	উদাহরণ	ফলাফল
+	যোগ করা	X = 15 + 6	X = 21
-	বিয়োগ করা	X = 15 - 6	X = 9
*	গুন করা	X = 15 * 6	X = 90
/	ভাগ করা	X = 15 / 6	X = 2
%	ভাগশেষ করা	X = 15 % 6	X = 3



6. Arithmetic operator(+, -, *, /)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int num1 = 10, num2 = 3;

    int sum = num1 + num2;
    cout << "Sum is: " << sum << endl;

    int sub = num1 - num2;
    cout << "subtraction is: " << sub << endl;

    int mul = num1 * num2;
    cout << "Multiplication is: " << mul << endl;

    double div = (double)num1 / num2;
    cout << "Division is: " << div << endl;

    int mod = num1 % num2;
    cout << "Modolus is: " << mod << endl;
}
```

Output:

Sum is: 13

subtraction is: 7

Multiplication is: 30

Division is: 3.33333

Modolus is: 1

Objectives

1. showpoint
2. noshowpoint
3. setprecision
4. fixed
5. setw()

7. Output Formating

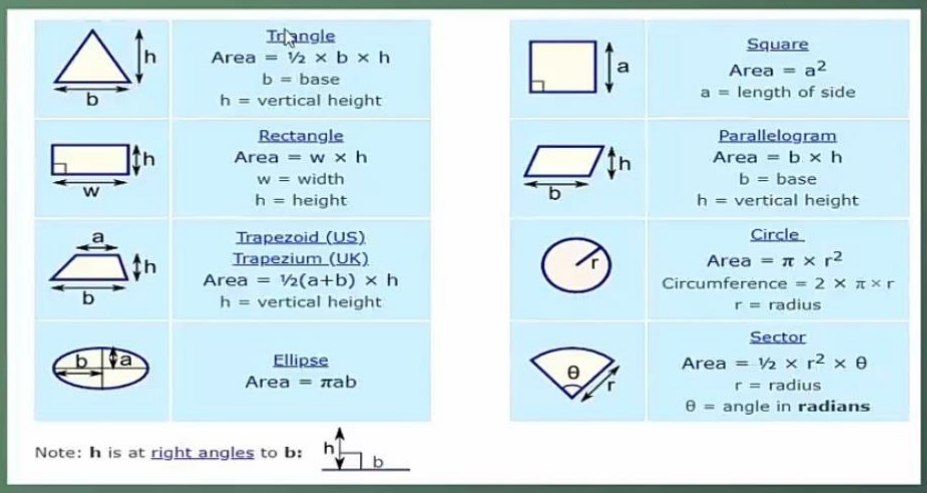
```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    double num1 = 10, num2 = 3;

    //Dosomik soho by default decimal point dekaby.
    cout << showpoint;
    //Sudu dosomik er por dui ghor dekaby.
    cout << fixed;
    //Dosomik soho dui ghor dekaby.
    cout << setprecision(2);

    double sum = num1 + num2;
    cout << setw(20) << "Sum is: " << sum << endl;

    //cout << noshowpoint;
```

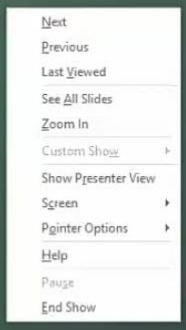
```
    double sub = num1 - num2;

    cout << setw(20) << "subtraction is: " << sub << endl;


    double mul = num1 * num2;

    cout << setw(20) << "Multiplication is: " << mul << endl;


    double div = (double)num1 / num2;

    cout << setw(20) << "Division is: " << div << endl;

}
```

Output:

```
            Sum is: 13.00
    subtraction is: 7.00
 Multiplication is: 30.00
       Division is: 3.33
```

Area of different shape





8. Area of Triangle.

```cpp
#include <iostream>
using namespace std;
int main()
{
    double base, height, area;
    cout << "Enter base and height ";
    cin >> base >> height;

    area = 1.0/2 * base * height;
    //area = (double)1/2 * base * height;
    //area = 0.5 * base * height;

    cout << "The area is: " << area;
}
```

Output:

```
Enter base and height 10 3
The area is: 15
```

Temperature Conversion Formulas

Equations for converting between Celsius (C), Fahrenheit (F), and Kelvin (K) temperature scales

$$°F = 1.8\ °C + 32 \quad \text{(i)}$$

$$°C = (°F - 32) / 1.8 \quad \text{(ii)}$$

$$K = °C + 273 \quad \text{(iii)}$$

9. Temperature conversion.

```cpp
#include <iostream>
using namespace std;
int main()
{
    double c, f;
    cout << "Enter celcious temp: ";
    cin >> c;

    f = 1.8 * c + 32;
    cout << "Temp in farenheight is: " << f;
}
```

Enter celcious temp: 32

Temp in farenheight is: 89.6

```cpp
#include <iostream>
using namespace std;
int main()
{
    double c, f;
    cout << "Enter farenheight temp: ";
    cin >> f;

    c = (f-32)/1.8;
    cout << "Temp in celcious is: " << c;
}
```

```
Enter farenheight temp: 89.6
Temp in celcious is: 32
```

Assignment Operator

Assignment Operator	Example	Full meaning
=	y = x + 5 ;	
+=	x += 5 ;	x = x + 5 ;
-=	x -= y ;	x = x - y ;
*=	x *= 5 ;	x = x * 5 ;
/=	x /= 5 ;	x = x / 5 ;
%=	x %= 5 ;	x = x % 5 ;

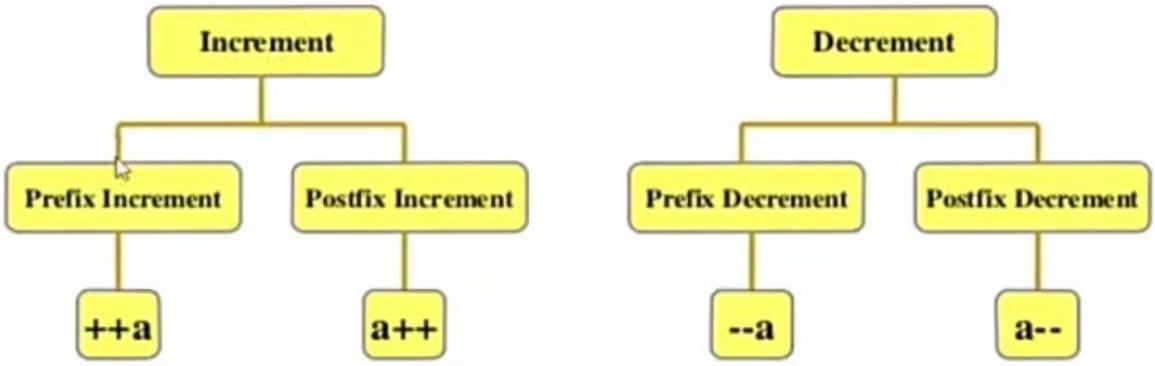
10. Assignment operator(+=, -+, *=, /+, %=)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 5, y = 3;

    x+=y;
    cout << "X = " << x << endl; //x = 8

    x-=y;
    cout << "X = " << x << endl; //x = 5

    x*=y;
    cout << "X = " << x << endl; //x = 15

    x/=y;
    cout << "X = " << x << endl; //x = 5

    x%=y;
    cout << "X = " << x << endl; //x = 2
}
```

Unary operator

Unary Operator	Meaning
+	Unary plus
-	Unary minus
++	Increment
--	Decrement

Increment and Decrement Operator



increment operator

```
++x;      // increments x by one - BEFORE it is used
x++;      // increments x by one - AFTER it is used
```

decrement operator

```
--x;      // decrements x by one - BEFORE it is used
x--;      // decrements x by one - AFTER it is used
```

11. Uniry operator(+, -, ++, --)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 3;
    int y = x++;
    cout << x << endl; //4
    cout << y << endl; //3

    int a = 5;
    int b = a--;
    cout << a << endl; //4
    cout << b << endl; //5

    int x1 = 3;
    int y1 = ++x1;
    cout << x1 << endl; //4
    cout << y1 << endl; //4
```

```
    int a1 = 5;
    int b1 = --a1;
    cout << a1 << endl; //4
    cout << b1 << endl; //4

    int m = 1;
    //(-) man return kory
    int n = -m;
    cout << n << endl; //-1

    int j = 1;
    //(+) man return kory
    int k = +j;
    cout << k << endl; //1
}
```

Bitwise operator

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise EXOR
>>	Bitwise Shift Right
<<	Bitwise Shift Left
~	Bitwise NOT

```
int a = 32;
int b = 12;
int c;

c = a & b;
cout << c ;
```

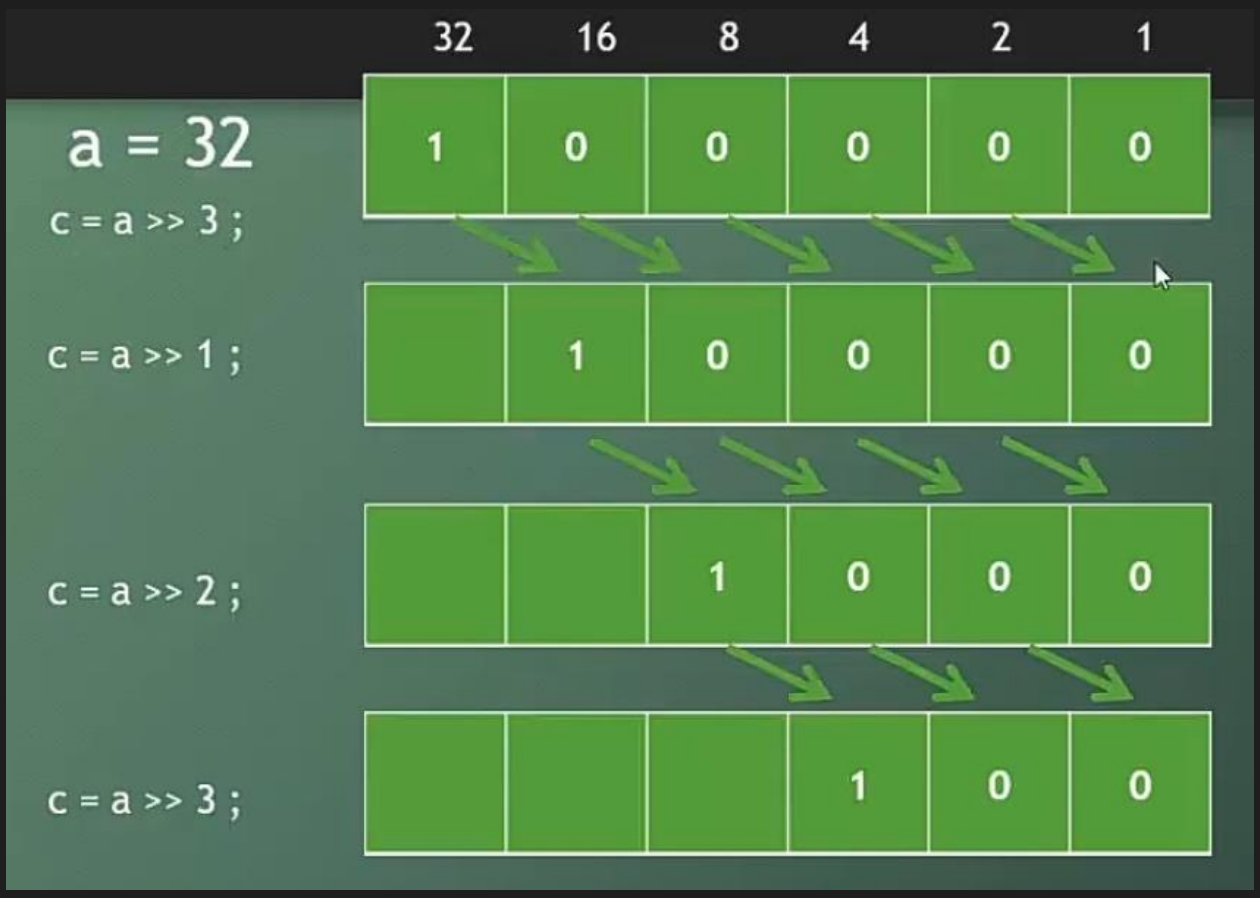
	32	16	8	4	2	1	
a = 32 =	1	0	0	0	0	0	
b = 12 =	0	0	1	1	0	0	
a & b =	0	0	0	0	0	0	→ 0

```
int a = 32;
int b = 12;
int c;

c = a ^ b;
cout << c ;
```

	32	16	8	4	2	1	
a = 32 =	1	0	0	0	0	0	
b = 12 =	0	0	1	1	0	0	
a ^ b =	1	0	1	1	0	0	→ 44

12. Bitwise operator(&, |, ^)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a = 10, b = 12, c;

    c = a & b;
    cout << c << endl; //8

    c = a | b;
    cout << c << endl; //14

    c = a ^ b;
    cout << c << endl; //6
}
//bijor songkok 1 thakly output o 1 hoby.
//details bujar jonno upor er pic gula deko.
```


32
16
8
4
2
1
a = 32
1 0 0 0 0 0
c = a >> 3 ;
c = a >> 1 ;
1 0 0 0 0
c = a >> 2 ;
1 0 0 0
c = a >> 3 ;
1 0 0

13. Bitwise operator(Left & Right shift)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a = 32, c;

    //orthat 32 k 3 bar 2 dara vag. 32/2=16/2=8/2=4
    c = a >> 3;
    cout << c << endl; //4

    //orthat 4 k 3 bar 2 dara gun. 4*2=8*2=16*2=32
    int x = 4, y;
    y = x << 3;
    cout << y << endl; //32
}
//details bujar jonno upor er pic gula deko.
```

Special operator

Comma (,)

Pointer (*)

sizeof()

14. Special operator(sizeof, comma)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int i;
    float f;
    double d;
    char c;
    char name[10];

    cout << sizeof(i) << endl; //4
    cout << sizeof(f) << endl; //4
    cout << sizeof(d) << endl; //8
    cout << sizeof(c) << endl; //1
    cout << sizeof(name) << endl; //10

    //comma er use. Bam pasy sub expression dan pasy purnaggo expression.
    int x, y, sum;
    sum = (x = 10, y = 5, sum = x + y);
    cout << sum << endl; //15
}
```

Relational operator

Operator	Use	Description
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

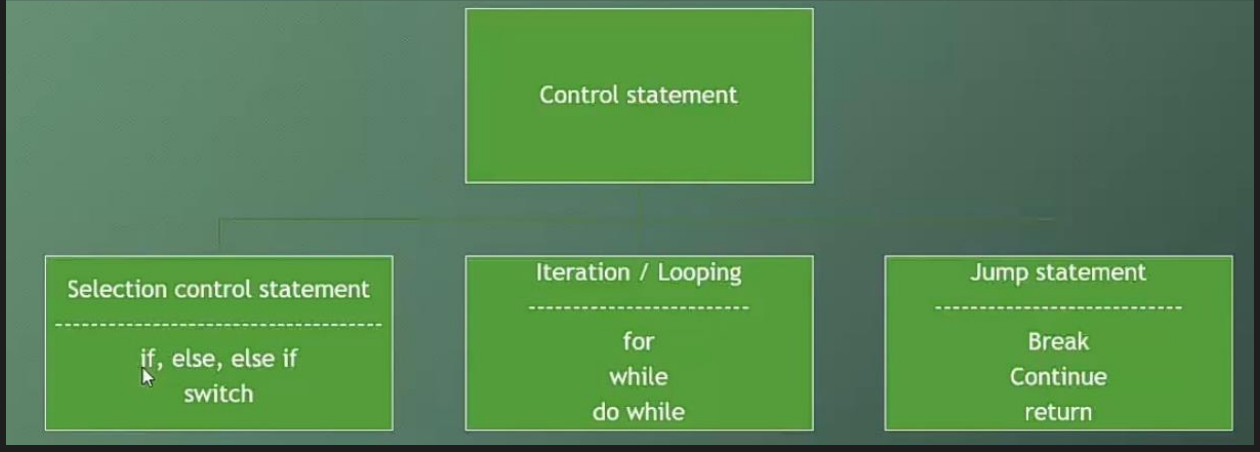
1. What is Statement?

- Any meaningful expression is called a statement.

Example

```
x=10;
cout << “Positive”;
cout<< “Negative”;
cout << “Zero”;
```

Remember a condition in if statement doesn't require a semicolon



15. Determine a Number is Positive or Negaive.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter an integer number: ";
    cin >> a;

    if (a > 0)
    {
        cout << "Positive" << endl;
    }
    if (a < 0)
    {
        cout << "Negative" << endl;
    }
    if (a == 0)
    {
        cout << "Zero" << endl;
    }
}
```

Or,

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter an integer number: ";
    cin >> a;

    if (a > 0)
    {
        cout << "Positive" << endl;
    }
    else if (a < 0)
    {
        cout << "Negative" << endl;
    }
    else if (a == 0)
    {
        cout << "Zero" << endl;
    }
}
```

Or,

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter an integer number: ";
    cin >> a;

    if (a > 0)
    {
        cout << "Positive" << endl;
    }
    else if (a < 0)
    {
        cout << "Negative" << endl;
    }
    else
    {
        cout << "Zero" << endl;
    }
}
```

16. Even or Odd

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter an integer number: ";
    cin >> a;

    if (a % 2 == 0)
    {
        cout << "Even" << endl;
    }
    else
    {
        cout << "Odd" << endl;
    }
}
```

17. Large number between two number.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout << "Enter two integer number: ";
    cin >> a >> b;

    if (a > b)
    {
        cout << "Large is: " << a << endl;
    }
    else
    {
        cout << "Large is: " << b << endl;
    }
}
```

18. Pass or Fail(33 marks easy)

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter number: ";
    cin >> a;

    if (a > 33)
    {
        cout << "Pass" << endl;
    }
    else
    {
        cout << "Fail" << endl;
    }
}
```

19. Absolute value print.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter number: ";
    cin >> a;

    if (a < 0)
    {
        cout << -a << endl;
    }
    else
    {
        cout << a << endl;
    }
}
```

20. Letter grade

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter your marks: ";
    cin >> a;

    if (a > 100 || a < 0)
        cout << "Invalid marks";
    else if (a >= 80 && a <= 100)
        cout << "A+";
    else if (a >= 70 && a <= 79)
        cout << "A";
    else
        cout << "Fail";
}
```

Logical operators

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

21. Logical operator(&&, ||)

```
#include <iostream>
using namespace std;
int main()
{
    int a = 12, b = 22;

    if (a > 20 && b < 25)
        cout << "Logical AND" << endl;
    if (a > 20 || b < 25)
        cout << "Logical OR" << endl;
}
```

Output:

Logical OR

22. Vowel or Consonant

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    cout << "Enter any character: ";
    cin >> c;
    c = tolower(c);
    if (c == 'a' || c == 'e' || c == 'i' || c == 'o'
|| c == 'u')
        cout << "Vowel" << endl;
    else
        cout << "Consonant" << endl;
}
//toupper();
```

23. Large number between three number.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int num1, num2, num3, large;
    cout << "Enter there integer number: ";
    cin >> num1 >> num2 >> num3;

    if(num1 > num2 && num1 > num3)
        large = num1;
    else if(num2 > num1 && num2 > num3)
        large = num2;
    else
        large = num3;

    cout << "Large number is : " << large << endl;
}
```

```
Output:
Enter there integer number: 32 21 11
Large number is : 32
```

Leap year Algorithm

START Step 1 → Take integer variable

year Step 2 → Assign value to the variable

Step 3 → Check if year is divisible by 4 but not 100, DISPLAY "leap year"

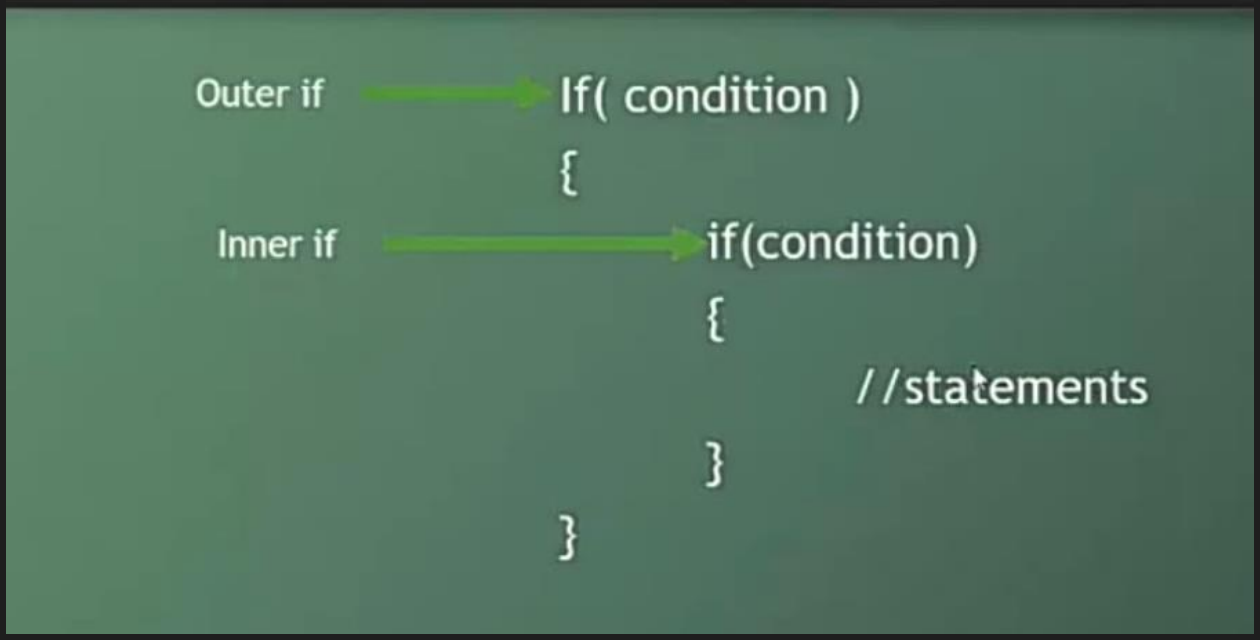
Step 4 → Check if year is divisible by 400, DISPLAY "leap year“

Step 5 → Otherwise, DISPLAY "not leap year" STOP

24. Leap Year.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int year;
    cout << "Enter a year: ";
    cin >> year;

    if (year % 4 == 0 && year % 100 != 0)
        cout << "Lear Year";
    else if (year % 400 == 0)
        cout << "Leap Year";
    else
        cout << "Not a Leap Year";
    return 0;
}
```

Nested if



25. Nested if

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter an Integer number: ";
    cin >> a;

    if (a > 32)
    {
        if (a >= 80 && a <= 100)
            cout << "A+";
        else if (a >= 70 && a <= 79)
            cout << "A";
    }
    else
    {
        cout << "Fail" << endl;
    }
}
```

26. Large number between two number using Ternary operator.

```
#include <iostream>
using namespace std;
int main()
{
    int num1 = 20, num2 = 40;

    int large = num1 > num2 ? num1 : num2;

    cout << large << endl;

    int a = 12;

    (a%2==0) ? cout << a << " is even" : cout << a << " is odd";
}
```

Control statement - switch

- Switch statement provides a better alternative than a large series of if-else statements.
- Switch statement executes one statement from multiple conditions.
- Keywords used in switch statement : switch, case, break, default

27. Switch statement.

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cout << "Enter a digit: ";
    cin >> a;

    switch (a)
    {
    case 0:
        cout << "Zero" << endl;
        break;
    case 1:
        cout << "One" << endl;
        break;
    case 2:
        cout << "Two" << endl;
        break;
```

```
        default:
            cout << "Not a digit. Plz try again!!";
            break;
        }
    }
```

28. Vowel or Consonant using Switch

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    cout << "Enter a character: ";
    cin >> c;
    c = tolower(c);
    switch (c) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        cout << "vowel" << endl;
        break;
     default:
        cout << "Consonant" << endl;
        break;
    }
}
```

What is loop? Why loop?

- A loop statement allows us to execute a statement or group of statements multiple times.

29. Simple for loop printing name five times.

```cpp
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i <= 5; i++)
    {
        cout << i << ". Golam kibria" << endl;
    }
}
```

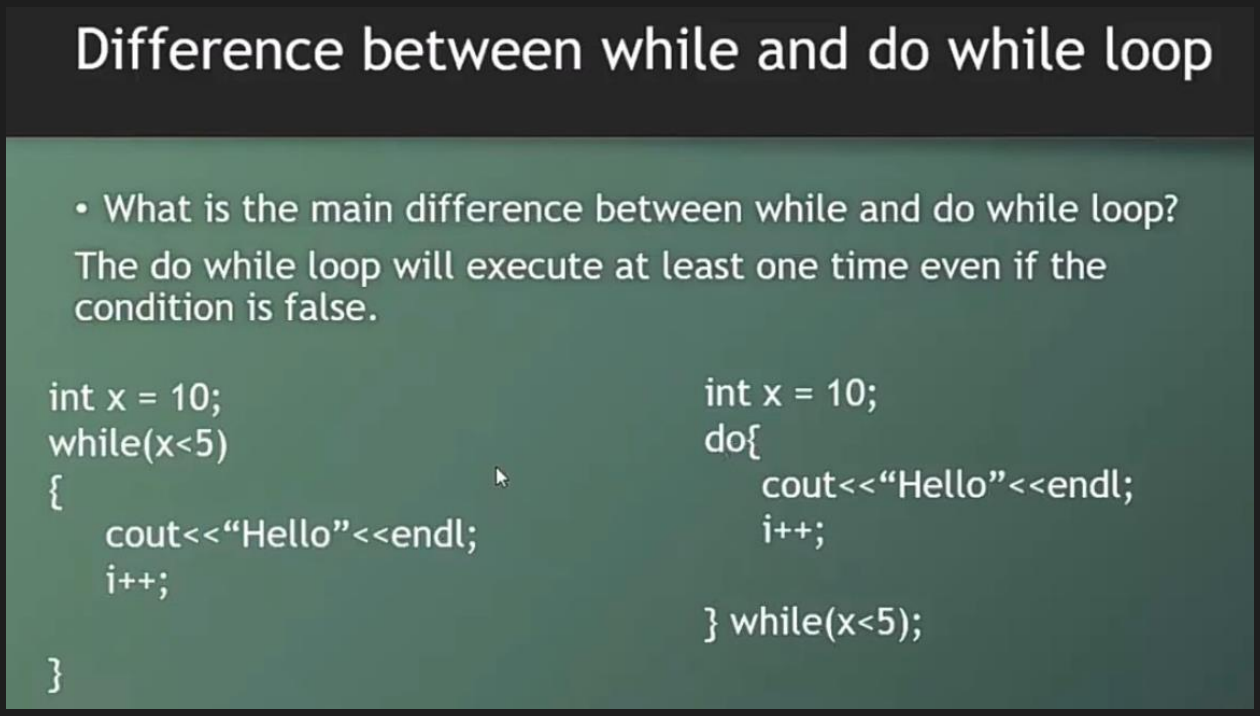
30. Simple while loop

```cpp
#include <iostream>
using namespace std;
int main()
{
    int i = 1;
    while (i <= 5)
    {
        cout << i << ". Bangladesh" << endl;
        i++;
    }
}
```

31. Simple do-while loop

```cpp
#include <iostream>
using namespace std;
int main()
{
    int i = 1;
    do
    {
        cout << i << ". Bangladesh" << endl;
        i++;
    } while (i <= 5);
}
```

32. Difference between while and do while loop.



33. break

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i < 20; i++)
    {
        if(i==10){
            break;
        }
        cout << i << endl;
    }
}
```

Output:

0 to 9

34. continue

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i <= 20; i++)
    {
        if (i == 10)
        {
            continue;
        }
        cout << i << endl;
    }
}
//direct loop a ferot ferot pataby. nichy kono
statement thakly ta execute hoby na.
Output:
0 to 20 except 10
```

35. break and continue

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i <= 20; i++)
    {
        if (i == 10)
        {
            continue;
        }
        if (i > 13)
        {
            break;
        }
        cout << i << endl;
    }
}
```

Output:

0 to 13 except 10.

Loop related programs

1. Multiplication Table
2. Factorial
3. Prime number
4. GCD & LCM
5. Sum of digits
6. Reverse number
7. Palindrome number
8. Armstrong Strong number
9. Strong number
10. Counting number of a digit in a number

36. Multiplication table

```cpp
#include <iostream>
using namespace std;
int main()
{
    int num;
    cout << "Enter a number: ";
    cin >> num;

    for (int i = 1; i <= 10; i++)
    {
      cout << num << "x" << i << " = " << (num * i) << endl;
    }
}
```

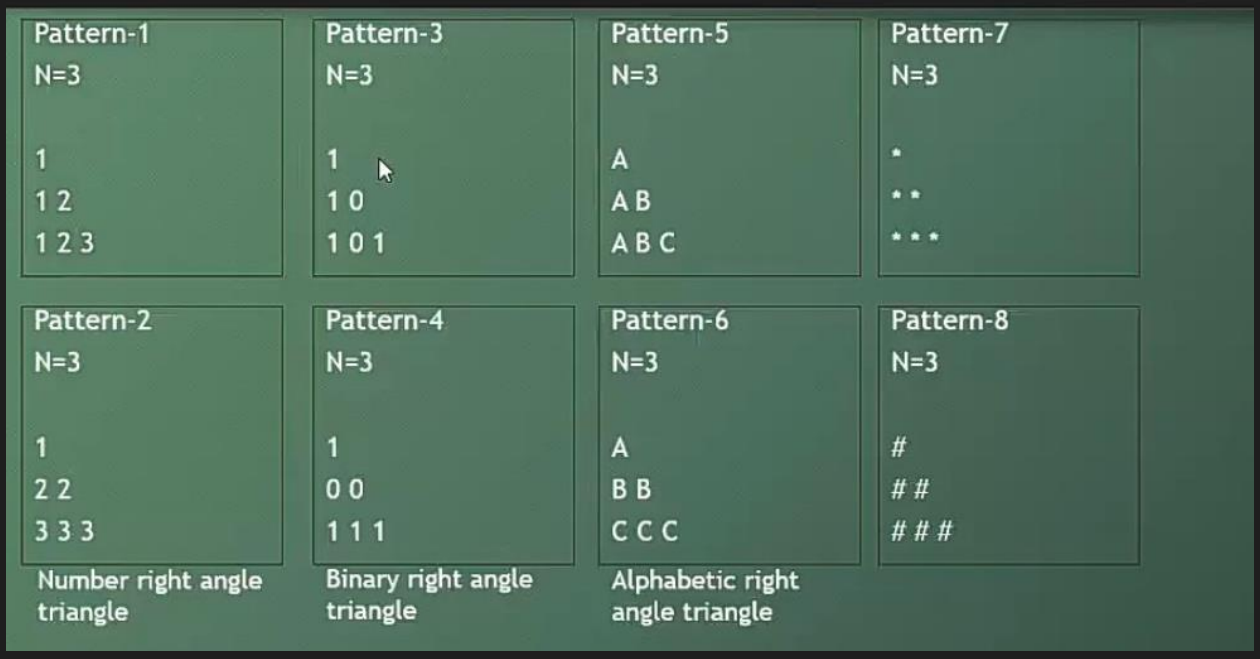
Objectives (series related programs)

1. 1+2+3+ … +n (Print sum of 1 to n)
2. 2+4+6+8+…+n (sum of even numbers from 1 to n)
3. 1+3+5+…+n (sum of odd numbers from 1 to n)
4. 1.5+2.5+3.5+…+n
5. 1+1/2+1/3+1/4+…+1/n
6. $1^2 + 2^2 + 3^2 + \ldots + n^2$
7. $1^5 + 2^5 + 3^5 + \ldots + n^5$
8. 1-2+3-4+5-6+……n
9. 1 X 2 X 3 X…. X n
10. 1^2 X 2^2 X 3^2 X…. X n^2
11. Fibonacci series
12. Lucas series

37. 1 + 2 + 3 + -------- + n

```cpp
#include <iostream>
using namespace std;
int main()
{
    int num, sum = 0;
    cout << "Enter the last number: ";
    cin >> num;

    for (int i = 1; i <= num; i++)
    {
        sum = sum + i;
    }
    cout << "The sum is: " << sum << endl;
}
/*
    Series er jonno 3 ta bisoy mathay rakty hoby.
    1. First number
    2. Last number
    3. Babodhan koto kory
*/
```

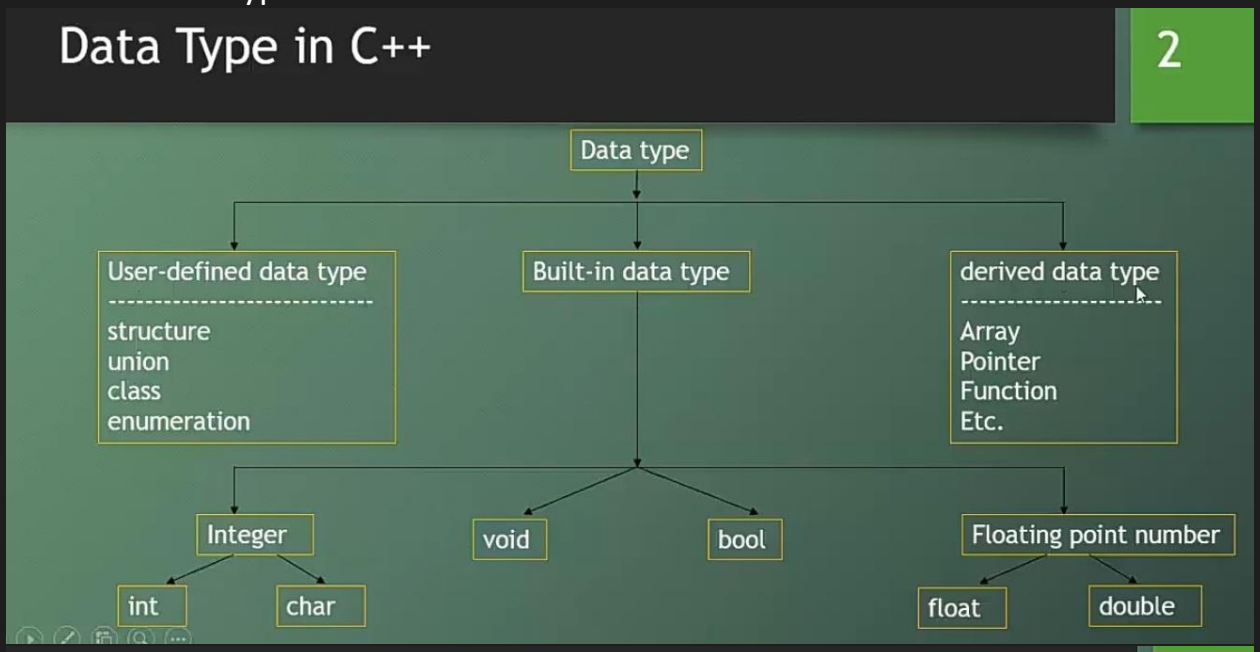
Pattern Type-1

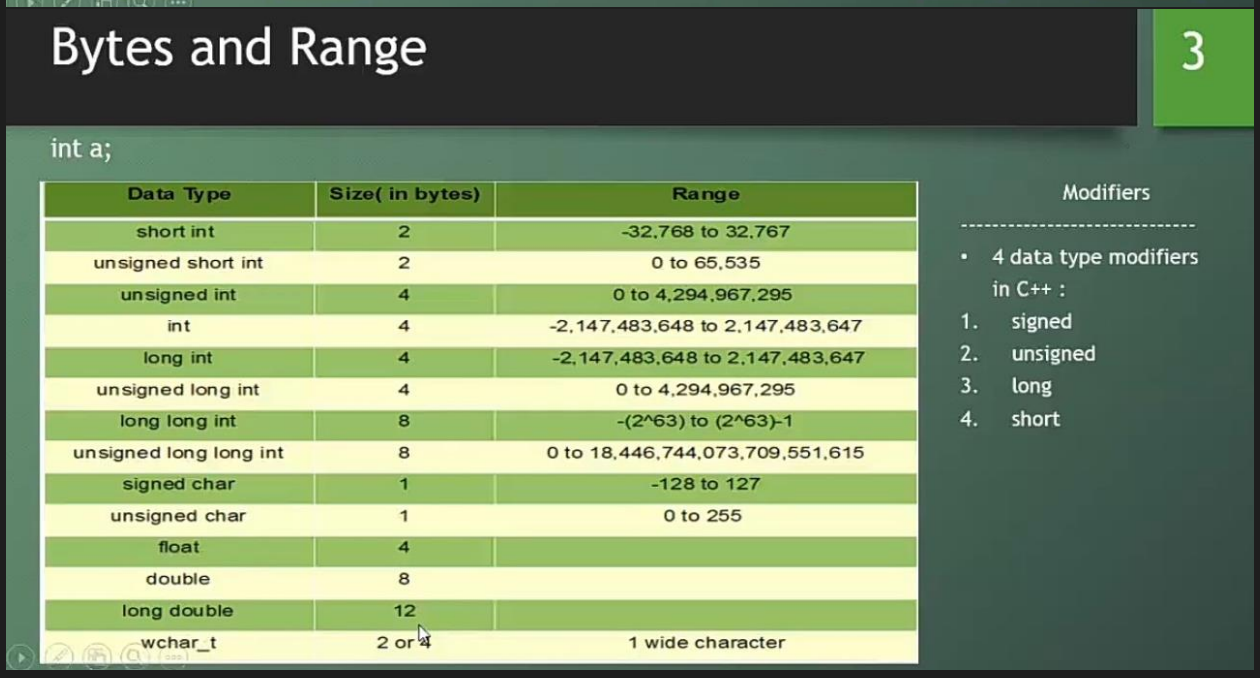


38. Basic pattern

```
#include <iostream>
using namespace std;
int main()
{
    int row, col, n;
    cout << "Enter how many lines: ";
    cin >> n;

    for (row = 1; row <= n; row++)
    {
        for (col = 1; col <= row; col++)
        {
            cout << col << " ";
        }
        cout << endl;
    }
}
1
1 2
1 2 3
```

39. Data type in C++





->int er jonno 4 byte. Tai er agy jodi long use kori taby seta digun hoye jaby. Orthat 8 byte hoye jaby. Thik temni short use korly ordek komy jaby.

->signed holy positive or negatice j kono value rakty parbo

->unsigned holy only positive value store korty parbo.

40. 1D array printing.

- Definition : An array is a collection of variables of same type.

```
#include <iostream>
using namespace std;
int main()
{
    int num[5] = {1, 2, 3, 4, 5};
    for(int i = 0; i<5; i++){
        cout << num[i] << " ";
    }
}
```

Output:

```
1 2 3 4 5
```

41. Take user input from an 1D array.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int num[5];
    for (int i = 0; i < 5; i++)
    {
        cout << "Students number " << i+1 << ":";
        cin >> num[i];
    }
    cout << "\nNumbers are: " << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << num[i] << " ";
    }
}
```

```
Output:
Students number 1:12
Students number 2:23
Students number 3:34
Students number 4:45
Students number 5:56

Numbers are:
12 23 34 45 56
```

42. Finding sum, avg, max & min.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n, sum = 0;
    cout << "Enter number of students: ";
    cin >> n;
    int student[n];

    cout << "Student Information" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "Value " << i + 1 << ": ";
        cin >> student[i];
        sum = sum + student[i];
    }
    cout << "Sum is: " << sum << endl;
    double avg = (double)sum / n;
    cout << "Average is: " << avg << endl;
```

```
    int max = student[0];
    int min = student[0];

    int j, position;
    for (j = 1; j < n; j++)
    {
        if (student[j] > max)
        {
            max = student[j];
            position = j;
        }
        if (student[j] < min)
        {
            min = student[j];
            position = j;
        }
    }
    cout << "Maximum number is: " << max << " which is index: " << position << endl;
    cout << "Minimum number is: " << min << endl;
}
```

43.Types of Array

Types of Array

- Arrays can of following types:

1. One dimensional (1-D) arrays or Linear arrays
   Example : int marks[10];
2. Multi dimensional arrays
   (a) Two dimensional (2-D) arrays or Matrix arrays
   Example : int marks[2][3];
   (b) Three dimensional arrays
   Example : int marks[2][3][2];

1D Array

1. What is Array?
2. Types of Array
3. Declaration, Initialization , sum of Array
4. Sum and Average of an Array
5. Maximum and Minimum of Array
6. Fibonacci series using array
7. Searching a number (Linear search)
8. copy all elements of an array to another array

44. Printing a simple 2D array.

```
#include <iostream>
using namespace std;
int main()
{
    int a[2][2] = {
        {3, 4},
        {5, 6}};
    int row, col;
    for (row = 0; row < 2; row++)
    {
        for (col = 0; col < 2; col++)
        {
            cout << a[row][col] << " ";
        }
        cout << endl;
    }
}
```

45. Taking user input from 2D array.

```
#include <iostream>
using namespace std;
int main()
{
    int a[2][2], row, col;
    cout << "Enter the element of the matrix." << endl;

    for (row = 0; row < 2; row++)
    {
        for (col = 0; col < 2; col++)
        {
            cout << "a[" << row << "]"
                 << "[" << col << "]"
                 << " = ";
            cin >> a[row][col];
        }
    }
```

```
    for (row = 0; row < 2; row++)
    {
        for (col = 0; col < 2; col++)
        {
            cout << a[row][col] << " ";
        }
        cout << endl;
    }
}
Enter the element of the matrix.
a[0][0] = 1
a[0][1] = 2
a[1][0] = 3
a[1][1] = 4
1 2
3 4
```

1. Introduction to 2D Array
2. Getting input for 2D Array
3. Simple Matrix
4. Matrix Addition & Subtraction
5. Matrix Multiplication
6. Matrix Transpose
7. Sum of diagonal elements of a matrix
8. Sum of upper & lower triangles elements

Introduction to pointer

```
int x = 5;
cout << x ;
cout << &x;
```

variable → x

value → 5

Variable address → 100

Pointer is a variable that stores/ points the address of another variable.

Declaration of pointer

Data_type *variable_name;

Example : int *p; char *ch;

2 symbols

& symbol is used to get the address of the variable.

* Symbol is used to get the value of the variable that the pointer is pointing to.

46. Pointer printing value and address.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x = 5;
    int *p;
    p = &x;

    cout << x << endl;
    cout << &x << endl;
    cout << p << endl;
    cout << *p << endl;
}
/* Sudu p er maddomy oi variabl er address ta print
kortechi. Ar *p er maddomy oi variable er value ta print
kortechi. */
```

```
5
0x61feb8
0x61feb8
5
```

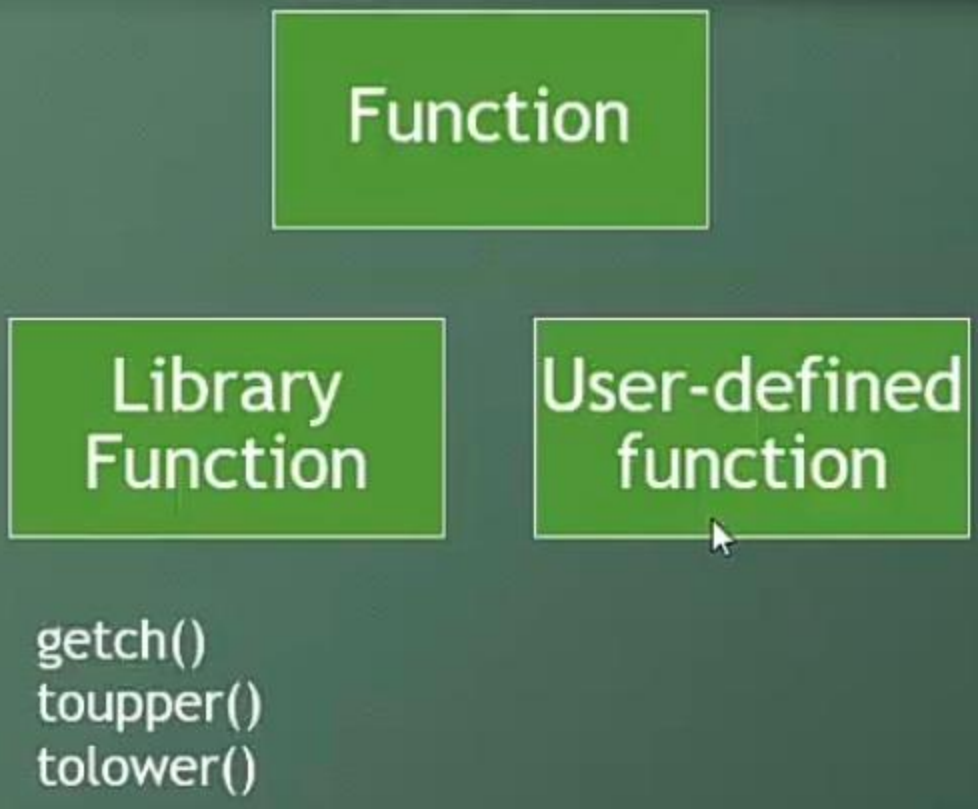
47. Printing sum of two number using pointer.

```
#include <iostream>
using namespace std;
int main()
{
    int num1 = 10, num2 = 20;
    int *p1 = &num1, *p2 = &num2;

    int sum = *p1 + *p2;
    cout << "SUM IS = " << sum << endl;
} or,
#include <iostream>
using namespace std;
int main()
{
    int num1 = 10, num2 = 20;
    int *p1, *p2;
    p1 = &num1;
    p2 = &num2;
    int sum = *p1 + *p2;
    cout << "SUM IS = " << sum << endl;
} // SUM IS = 30
```

Function

- A **function** is a group of statements that perform a particular task.
- If you need to do same thing again and again then you may use function.

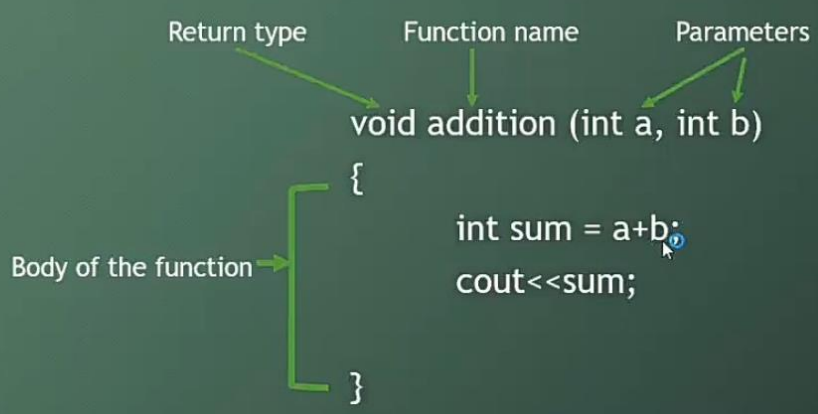
```
int main()
{
        //group of statements
}
```



1. Code Reusability
2. can use the same function for different inputs.

How to declare a Function

```
int main()
{
        int x = 10, y = 20;
        int sum = x+y;
        cout<<sum;

        x = 20, y = 20;
        sum = x+y;
        cout<<sum;

        x = 20, y = 30;
        sum = x+y;
        cout<<sum;
}
```



48. Adding two number using function.

```cpp
#include <iostream>
using namespace std;
void add(int a, int b)
{
    int sum = a + b;
    cout << sum << endl;
}
int main() {
    add(10, 20);
} or,
#include <iostream>
using namespace std;
void add()
{
    int a = 10, b = 20;
    int sum = a + b;
    cout << sum << endl;
}
int main(){
    add();
} //30
```

49. Add, sub, div, multi using function.

```cpp
#include <iostream>
using namespace std;
void add(int a, int b){
    int sum = a + b;
    cout << sum << endl;
}
void sub(int a, int b){
    int subt = a - b;
    cout << subt << endl;
}
void div(double a, double b){
    double division = a / b;
    cout << division << endl;
}
void multi(int a, int b){
    int cross = a * b;
    cout << cross << endl;
}
```

```c
int main()
{
    add(10, 20);
    sub(20, 10);
    div(10.5, 2.2);
    multi(2, 3);
}
```

50. Return a value from a function.

```cpp
#include <iostream>
using namespace std;
int add(int a, int b){
    int sum = a + b;
    return sum;
}
int main()
{
    int result = add(10, 5);
    cout << result << endl;
}
```

51. Default parameter value.

```cpp
#include <iostream>
using namespace std;
void display(int a = 10, int b = 20)
{
    cout << a << endl;
    cout << b << endl;
}
int main()
{
    display();
    display(5); // a er value 5 hoye jaby.
}
```

Output:

```
10
20
5
20
```

52. Function without parameter.

```cpp
#include <iostream>
using namespace std;
void square()
{
    int a = 5;
    int result = a * a;
    cout << result << endl;
}
int main()
{
    square();
}
```

Output: 25

Or,

```
//nicher ta korychi cuz parameter thakly subida hoy.
```

```cpp
#include <iostream>
using namespace std;
void square(int a)
{
    int result = a * a;
    cout << result << endl;
}
int main()
{
    square(5);
    square(6);
    square(7);
}
```

53. Random number print.

```cpp
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
    for (int i = 0; i < 5; i++)
    {
        int rNumber = rand() % 5 + 1;
        cout << "Random number = " << rNumber << endl;
    }
}
//% 5 dara bujay random number gulo 0 hoty 4 er moddy asby. amra 0 chai na tai 1 jok kory diyechi.
```

54. Guessing game.

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
    int count = 0;
    while (1)
    {
        count = count + 1;
        int guessNumber;
        cout << "Enter your guess number: ";
        cin >> guessNumber;

        int rNumber = rand() % 5 + 1;

        if (guessNumber == rNumber)
        {
            cout << "You won" << endl;
        }
```

```
        else
        {
            cout << "You lost. Try again. " << endl;
            cout << "Random number = " << rNumber << endl;
        }
        if(count == 6){
            break;
        }
    }
}
//beak use korychi loop hoty ber hoyar jonno.
```

What is function overloading ?

Function overloading is a process of declaring

1. multiple functions with the **same name**
2. **Different parameters**

55. Function overloading.

```cpp
#include <iostream>
using namespace std;
void sum(int a, int b){
    int result = a + b;
    cout << "Sum is : " << result << endl;
}
void sum(int a, int b, int c){
    int result = a + b + c;
    cout << "Sum is : " << result << endl;
}
int main()
{
    sum(10, 20);
    sum(10, 20, 30);
}
```

56. Passing array through a function.

```cpp
#include <iostream>
using namespace std;
void display(int number[], int aSize)
{
    for (int i = 0; i < aSize; i++)
    {
        cout << number[i] << " ";
    }
}
int main()
{
    int number[5] = {1, 2, 3, 4, 5};
    display(number, 5);
}
/* Function a array pass koranor jonno parameter hisaby dui
ta value dity hoby. 1-ArrayName, 2-ArraySize. That's it */
```

Output: 1 2 3 4 5

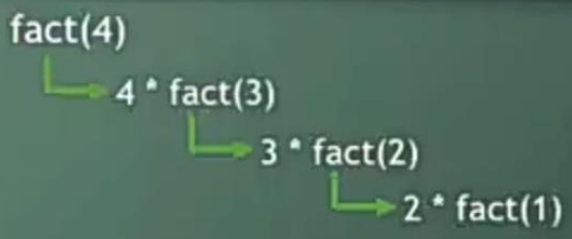
Recursion

- Recursion is a process where a function can call itself.
- To stop calling we need a base case.

2 important points in case of Recursion

1. Recursive call
2. Base case

```
int fact (int n)
{
        //base case
        if(n==1)
                return 1;

        else
                return n * fact(n-1);
}
```



57. Factorial using recursion.

```cpp
#include <iostream>
using namespace std;
int fact(int n)
{
    if(n == 1){
        return 1;
    } else{
        return n * fact(n -1);
    }
}
int main()
{
    int result = fact(4);
    cout << "Factorial is = " << result << endl;
}
```

Output:

Factorial is = 24

/* Bujar jonno uppor er pic ta deko */

Passing arguments to a function

- There are 2 ways to pass value into a function.

1. Pass by value
2. Pass by reference

Pass by value

In case of pass by value, a copy of the argument is passed to the function

Actual parameter / Argument

Formal parameter

```
int main()
{
    int x = 10;
    cout << "Before calling the function x = "<<x<<endl;

    display(x);
    cout << "After calling the function x = "<<x<<endl;

    getch();
}
```

```
void display(int num)
{
    num = 20;
}
```

58. Call by value

```
#include <iostream>
using namespace std;
void display(int num){
    num = 20;
}
int main()
{
    int x = 10;
    cout << "Before calling the function x = " << x << endl;

    display(x);
    cout << "After calling the function x = " << x << endl;
}
/*Orthat man er kono poriborton hoy nai. Karon display
function a amra sudu x er akta copy patiyechi foly original
valuer kono change hoy nai. */
```

```
Before calling the function x = 10
After calling the function x = 10
```

Pass by reference

In case of pass by reference, copies an argument's address into the formal parameter.

Actual parameter

Formal parameter

```
int main()
{
    int x = 10;
    cout << "Before calling the function x = "<<x<<endl;

    display(&x);
    cout << "After calling the function x = "<<x<<endl;

    getch();
}
```

```
void display(int *num)
{
    *num = 20;
}
```

59. Call by ruference.

```cpp
#include <iostream>
using namespace std;
void display(int *num){
    *num = 20;
}
int main()
{
    int x = 10;
    cout << "Before calling the function x = " << x << endl;

    display(&x);
    cout << "After calling the function x = " << x << endl;
}
/* Aykhetry kintu value change hoye jaccy. Karon ki? karon holo amra &x er moddymoy x er address taky display function a pattachi then *num er maddomy oi address hoty value ta receive kortechi torpor tar man change kory disi. Jar karony value ta change hoye giyechy. */
```

```
Before calling the function x = 10
After calling the function x = 20
```

60. Taking string input using gets()

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    char array[20];
    cout << "Enter your name: ";
    gets(array);

    cout << "Welcome " << array;
}
Welcome Golam kibria
```

61. Scope resolution operator.

```
#include <iostream>
using namespace std;
int x = 20;
void display(){
    cout << "Inside the display function x = " << x << endl;
}
int main()
{
    cout << "Inside the main function x = " << x << endl;
    display();
}//dui khetrei output 20
#include <iostream>
using namespace std;
int x = 20;

int main()
{
    int x = 10;
    cout << x;
}//output 10 karon main function er moddy x holo main
```

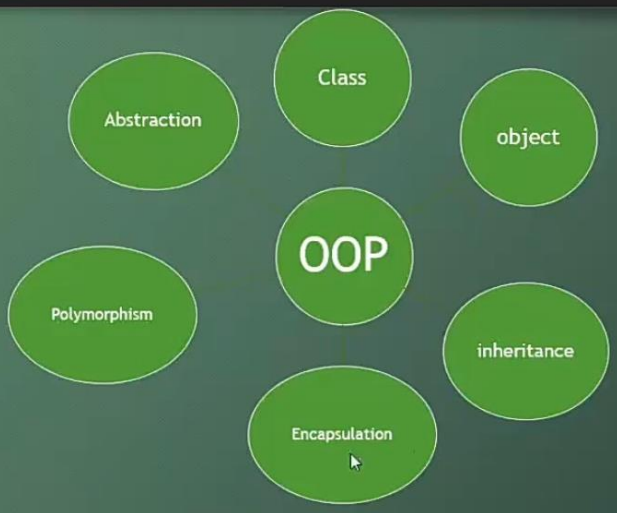
er local variable tai er power global hoty besi.

```cpp
#include <iostream>
using namespace std;
int x = 20;
int main()
{
    int x = 10;
    cout << ::x;
}//output 20

#include <iostream>
using namespace std;
int x = 20;

int main()
{
    int x = 10;
    :: x = 30;
    cout << ::x;
}//output 30
```

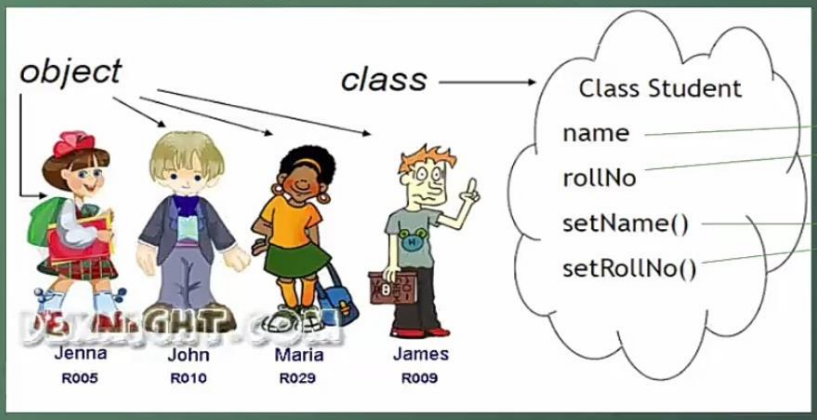
62. Understanding class & object

Object oriented programming concepts

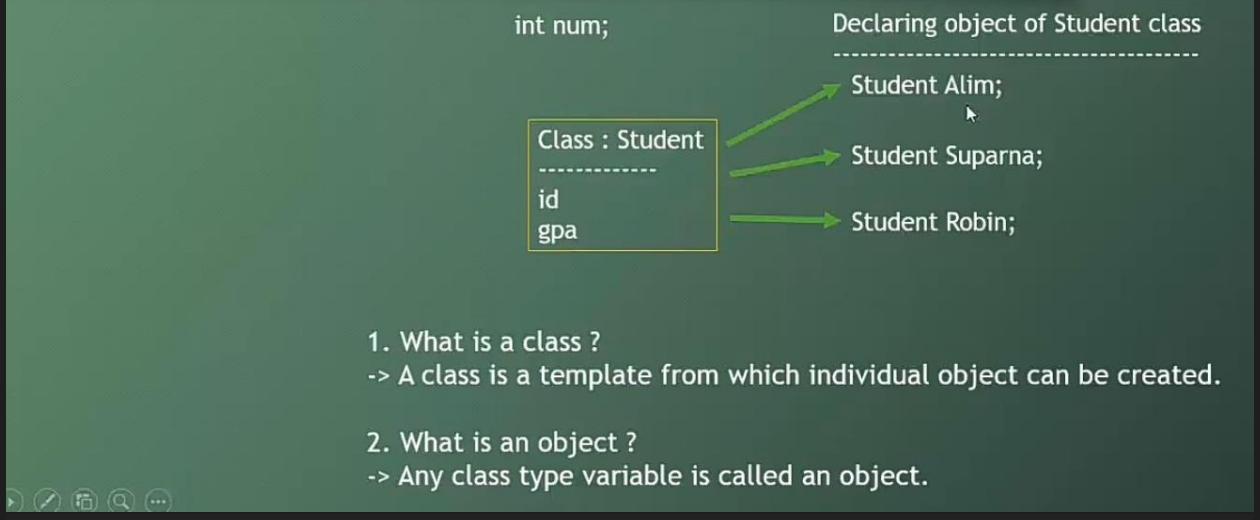
C + OOP = C++



Class and Object



Understanding class & object



1. What is a class ?
-> A class is a template from which individual object can be created.

2. What is an object ?
-> Any class type variable is called an object.

class declaration

```
class className
{
        public :
        //variables
        //functions
};
```

```
class Student
{
        public :
                int id;
                double gpa;

};
```

Object declaration

Class declaration

```
class Student
{
    public :
        int id;
        double gpa;

};
```

Object declaration

```
className objectName;
```

Example

```
Student Atik;
Student Suparna;
Student Robin;

Student Atik,Suparna,Robin;
```

63. Class & object create example.

```cpp
#include <iostream>
using namespace std;
class Student
{
public:
    int id;
    double gpa;
};
int main()
{
    Student Kibria, Munna;
    Kibria.id = 10;
    Kibria.gpa = 3.50;
    cout << Kibria.id << " " << Kibria.gpa << endl;

    Munna.id = 11;
    Munna.gpa = 3.60;
    cout << Munna.id << " " << Munna.gpa << endl;
}
```

```
10 3.5
11 3.6
```

64. Using function inside the class.

```cpp
#include <iostream>
using namespace std;
class Student
{
public:
    int id;
    double gpa;

    void display(){
        cout << id << " " << gpa << endl;
    }
};
```

```
int main()
{
    Student Kibria, Munna;
    Kibria.id = 10;
    Kibria.gpa = 3.50;
    Kibria.display();

    Munna.id = 11;
    Munna.gpa = 3.60;
    Munna.display();
}
```

Output:

```
10 3.5
11 3.6
```

65. Using parameterized function inside the class

```
#include <iostream>
using namespace std;
class Student
{
public:
    int id;
    double gpa;

    void setValue(int x, double y){
        id = x;
        gpa = y;
    }
    void display(){
        cout << id << " " << gpa << endl;
    }
};
```

```
int main()
{
    Student Kibria, Munna;
    Kibria.setValue(10, 3.50);
    Kibria.display();
    Munna.setValue(11, 3.60);
    Munna.display();
}
```

Output:

```
10 3.5
11 3.6
```

66. Constructor example.

Constructor

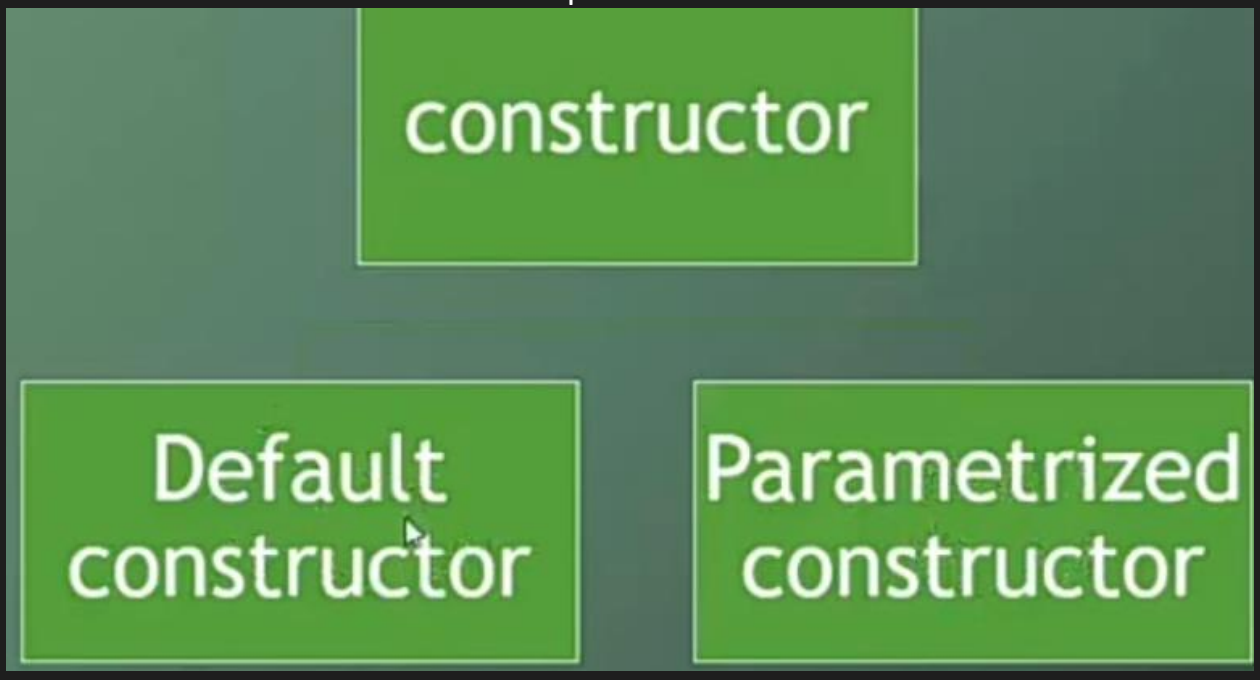
- What is a constructor ?
- Constructor is a special type of function that is used to initialize the object.

- What are the properties of constructor ?

1. Constructor is a special type of function.
2. Constructor has the same name as that of the class it belongs.
3. It has no return type not even void.
4. It is called automatically.

```cpp
#include <iostream>
using namespace std;
class Student
{
public:
    int id;
    double gpa;

    Student(int x, double y){
        id = x;
        gpa = y;
    }
    void display(){
        cout << id << " " << gpa << endl;
    }
};
int main() {
    Student Kibria(10, 3.50);
    Kibria.display();
    Student Munna(11, 3.60);
    Munna.display();
}//same output ager tar moto.
```

67. Default constructor with example.



```cpp
#include <iostream>
using namespace std;
class Student
{
public:
    int id;
    double gpa;

    Student(int x, double y){
        id = x;
        gpa = y;
    }
    Student(){
        cout << "I'm default constructor" << endl;
    }
    void display(){
        cout << id << " " << gpa << endl;
    }
};//class end
```

```
int main()
{
    Student ob;
    Student Kibria(10, 3.50);
    Kibria.display();
    Student Munna(11, 3.60);
    Munna.display();
}
```

Output:

```
I'm default constructor
10 3.5
11 3.6
```

68. String details with example

1. What is a string ?

- A String is a sequence of characters.

- Example

  “This is a string.”

  “a”

  “ ”

String representation

- There are 2 ways of string representation in c++.

1. The c style character string.
2. string class

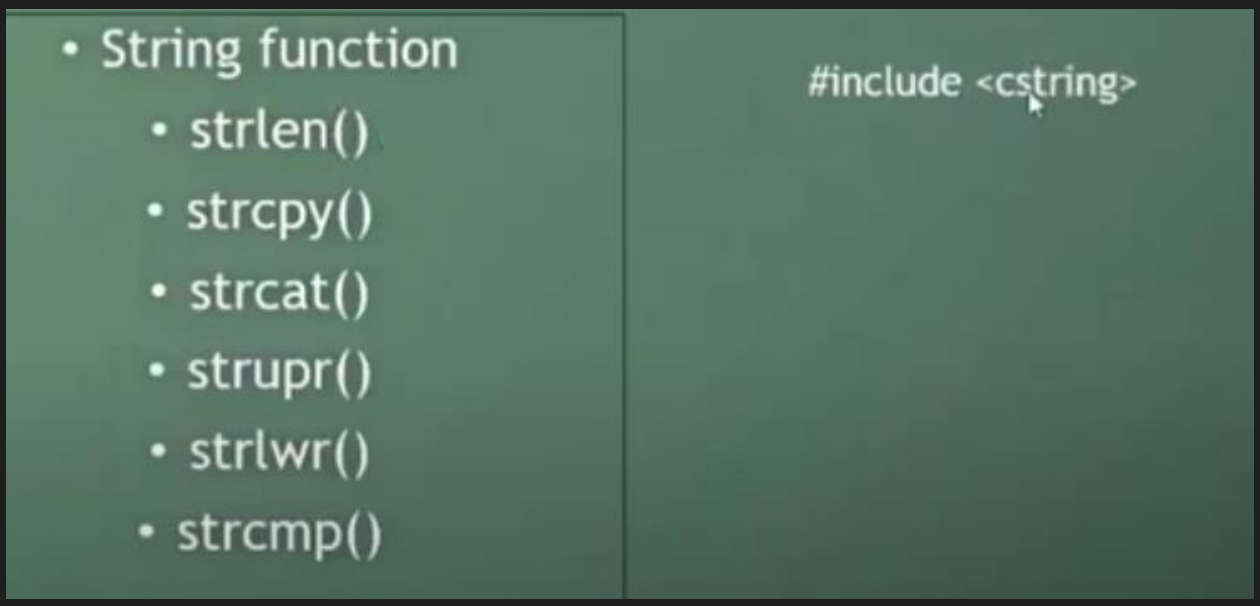
1. The c style character string.

```
char message[6] = { 'h', 'e', 'l', 'l', 'o' } ;

char message[] = { 'h', 'e', 'l', 'l', 'o', '\0' } ;

char message[] = "hello" ;
```

```cpp
#include<iostream>
#include<stdio.h>
using namespace std;

int main()
{
    char array[5] = {'a', 'b', 'c', 'd'};
    cout << array << endl;

    char array2[5] = {'q', 'e', 'r', 't'};
    cout << array2[2] << endl;

    char array3[] = {'q', 'e', 'r', 't', '\0'};
    cout << array3 << endl;

    //double quotion er moddy direct stirg declare kory diyechi.
    char array4[] = "Golam kibria";
    cout << array4 << endl;

    char name[20];
    cout << "Enter your name: ";
    //cin >> name;
    gets(name);
    cout << "Welcome " << name << endl;
}
```

```
Output:
abcd
r
qert
Golam kibria
Enter your name: Golam kibria
Welcome Golam kibria
```

69. String library function.



Example:

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

int main()
{
    char name1[] = "Golam";
    int length = strlen(name1);
    cout << "Length is : " << length << endl;

    char name2[20];
    strcpy(name2, name1);
    cout << "Name2 is : " << name2 << endl;

    char name3[] = "Abdur";
    char name4[] = " Rahim";
    strcat(name3, name4);
    cout << name3 << endl;

    char name5[] = "MUNNA";
    strlwr(name5);
    cout << name5 << endl;

    char name6[] = "munna";
    strupr(name6);
    cout << name6 << endl;
```

```
    char name7[] = "Kibria";
    char name8[] = "Golam";
    int value = strcmp(name7, name8);
    if(value == 0){
        cout << "String are equal" << endl;
    } else{
        cout << "String are not equal" << endl;
    }
}
```

Output:

```
Length is : 5
Name2 is : Golam
Abdur Rahim
munna
MUNNA
String are not equal
```

70.String class(add str & finding size)

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

int main()
{
    string str1 = "Golam";
    string str2 = " Kibria";
    string str3;

    str3 = str1;
    cout << "str3 : " << str3 << endl;

    str3 = str1 + str2;
    cout << "str3 : " << str3 << endl;

    int length = str1.size();
    cout << "Size is : " << length << endl;
}
```

71. Creating separate files for class

```
#include <iostream>
#include "myfirstclass.h"
using namespace std;

int main()
{
    MyFirstClass ob;
    ob.display();
    return 0;
}
#ifndef MYFIRSTCLASS_H
#define MYFIRSTCLASS_H


class MyFirstClass
{
    public:
        MyFirstClass();
        void display();
};

#endif // MYFIRSTCLASS_H

#include "myfirstclass.h"
#include<iostream>
using namespace std;
```

```
MyFirstClass::MyFirstClass()
{
    cout << "I'am constructor" << endl;
}
void MyFirstClass::display()
{
    cout << "Inside the display function" << endl;
}
```

Output:

```
I'am constructor
Inside the display function
```

72. Destructor

```
#include <iostream>
#include "myfirstclass.h"
using namespace std;

int main()
{
    MyFirstClass ob;
    ob.display();
    return 0;
}
```

```
#ifndef MYFIRSTCLASS_H
#define MYFIRSTCLASS_H


class MyFirstClass
{
    public:
        MyFirstClass();
        ~MyFirstClass(); //tild sign
        void display();
};
#endif // MYFIRSTCLASS_H
```

```
#include "myfirstclass.h"
#include<iostream>
using namespace std;
```

```
MyFirstClass::MyFirstClass()
{
    cout << "I'am constructor" << endl;
}
MyFirstClass::~MyFirstClass()
{
    cout << "I'am destructor" << endl;
}
void MyFirstClass::display()
{
    cout << "Inside the display function" << endl;
}
```

Output:

```
I'am constructor
Inside the display function
I'am destructor
/* Deka jaccy destructor sobar sesy print hoyechy karon object kono
resource use korly ta sobar sesy destroy kory deya hoy */
```

73. Selection operator

```
#include <iostream>
#include "myfirstclass.h"
using namespace std;

int main()
{
    MyFirstClass ob;
    MyFirstClass *p = &ob;
    p->display(); //-> selection operator.
    return 0;
}
//MyFirstClass er object er address k pointer er moddy reky dicci.

#ifndef MYFIRSTCLASS_H
#define MYFIRSTCLASS_H


class MyFirstClass
{
    public:
        void display();
};

#endif // MYFIRSTCLASS_H

#include "myfirstclass.h"
#include<iostream>
```

```
using namespace std;

void MyFirstClass::display()
{
    cout << "Inside the display function" << endl;
}
```

Output:

```
Inside the display function
```

74. Consonant variable

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

int main() {
    const int x = 20;
    cout << x << endl;

    return 0;
}
Output: 20
//many aytar value r change korty parbo na. akbay fixed.
```

75. Consonant object

```
#include <iostream>
#include "demo.h"
using namespace std;

int main()
{
    const Demo ob;
    ob.display();
    /*Orthat contant object er maddomy non-constant k call kora jaby na*/
    Demo ob2;
    ob2.display2();
    return 0;
}
```

```
#ifndef DEMO_H
#define DEMO_H


class Demo
{
public:
    void display() const;
    void display2();
};
#endif // DEMO_H
```

```
#include "demo.h"
```

```
#include <iostream>

using namespace std;
//class name
void Demo :: display() const
{
    cout << "I'am constant function" << endl;
}
void Demo :: display2()
{
    cout << "I'am non-constant function" << endl;
}
Output:
I'am constant function
I'am non-constant function
```

76. Constructor initializer

```
#include<iostream>
using namespace std;
class Student
{
public:
    const int age;
    const int fees;
    int id;
    Student(int x, int y, int z)
    //ay line tai holo constructor initializer.
    : age(x), fees(y)
    {
        cout << age << endl;
        cout << fees << endl;
        id = z;
        cout << id << endl;
    }
};
int main()
{
    Student ob(20, 2000, 10);
    return 0;
}
```

Output:

```
20
2000
10
```

- There are 4 core concepts in OOP.

Access specifiers in c++

- Access specifiers in c++ .

1. public
2. private
3. protected

```
class EncapTest {
        public :
                int roll;
                void display()
                {
                  .................
                }
        private :
                string name;
        protected :
                float gpa;

};
```

Encapsulation

Encapsulation is a process of

1. combining variables and functions in a single unit (class)

```
class Student {
    public :
        int id;
        string name;

        void display ()
        {
         cout << id << endl;
         cout << name <<endl;
        }

};
```

```
int main()
{
    Student s1;
    s1.id = 101;
    s1.name = "Suparna";
    s1.display();
    getch();

}
```

Encapsulation is a process of

1. combining variables and functions in a single unit (class)
2. Protecting data by declaring them as private

```
class Student {
    private :
        int id;
        string name;
    public :
        void display ()
        {
         cout << id << endl;
         cout << name <<endl;
        }

};
```

Private data will be hidden from other classes and they can only be accessed through public function of their current class. That is known as data hiding.

77. Setter and getter.

```cpp
#include <iostream>
using namespace std;
class Student
{
private:
    string name;
public:
    void setName(string x){
        name = x;
    }
    string getName(){
        return name;
    }
};

int main()
{
    Student s;
    s.setName("Kibria");
    cout << "Name is : " << s.getName();
} //Name is : Kibria
```

78. This keyword

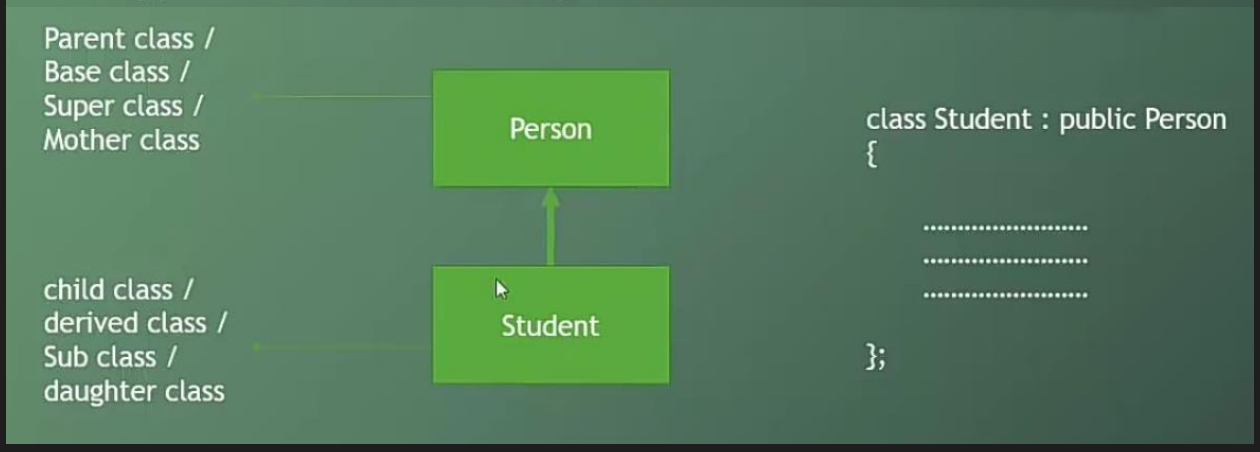
```cpp
#include <iostream>
using namespace std;
class Student
{
public:
    string name;

    Student(string name){
        this -> name = name;
    }
    void display(){
        cout << name << endl;
    }
};

int main()
{
    Student s("kibria");
    s.display();
}//kibria
```

79. Intheritance theory and example.

1. What is Inheritance ?

   The process of obtaining the data members and functions from one class to another class is known as **inheritance**.

2. What are the Importance of inheritance ?
   - ✓ Code reusability
   - ✓ Application development time is less.
   - ✓ Application take less memory.



```cpp
#include <iostream>
using namespace std;
class Person
{
public:
    string name;
    int age;
    void display1(){
        cout << "Name : " << name << endl;
        cout << "Age : " << age << endl;
    }
};
class Student : public Person
{
public:
    int id;
    //name, age, display1 auto choly aschy.
    void display2(){
        cout << "ID : " << id << endl;
        display1();
    }
};
```

```
int main()
{
    Student s;
    s.id = 10;
    s.name = "kibria";
    s.age = 20;
    s.display2();
}
```

80. Types of inheritance.

Types of inheritance

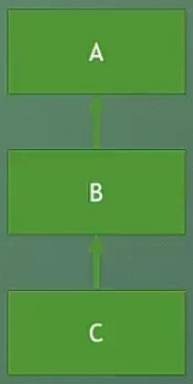
- There are 5 types of inheritance in c++.
  1) Single inheritance
  2) Multilevel inheritance
  3) Hierarchical inheritance
  4) Multiple inheritance
  5) Hybrid inheritance

1) Single inheritance

- In this sort of inheritance, one subclass inherit from one superclass.

```
class A
{
        .......
        .......
};

class B : public A
{
        .......
        .......
};
```

2) Multilevel inheritance



- In this sort of inheritance, the superclass for one is the subclass for Another.

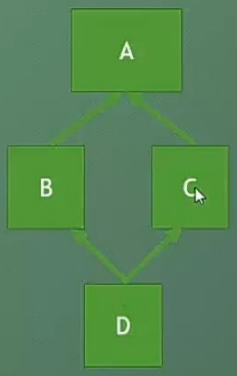
```
class A
{
        .......
        .......
};
class B : public A
{
        .......
        .......
};
class C : public B
{
        .......
        .......
};
```

3) Hierarchical inheritance

- In this sort of inheritance, multiple subclass derived from single superclass.

```
class A
{
        .....
        .....
};
class B : public A
{
        .....
        .....
};
class C : public A
{
        .....
        .....
};
```

4) Multiple inheritance



In multiple inheritance, a class can inherit more than one class.

in this type of inheritance a single child class can have multiple parent classes.

5) Hybrid Inheritance

- Hybrid inheritance is a combination of more than one type of inheritance.

81. Function overriding.

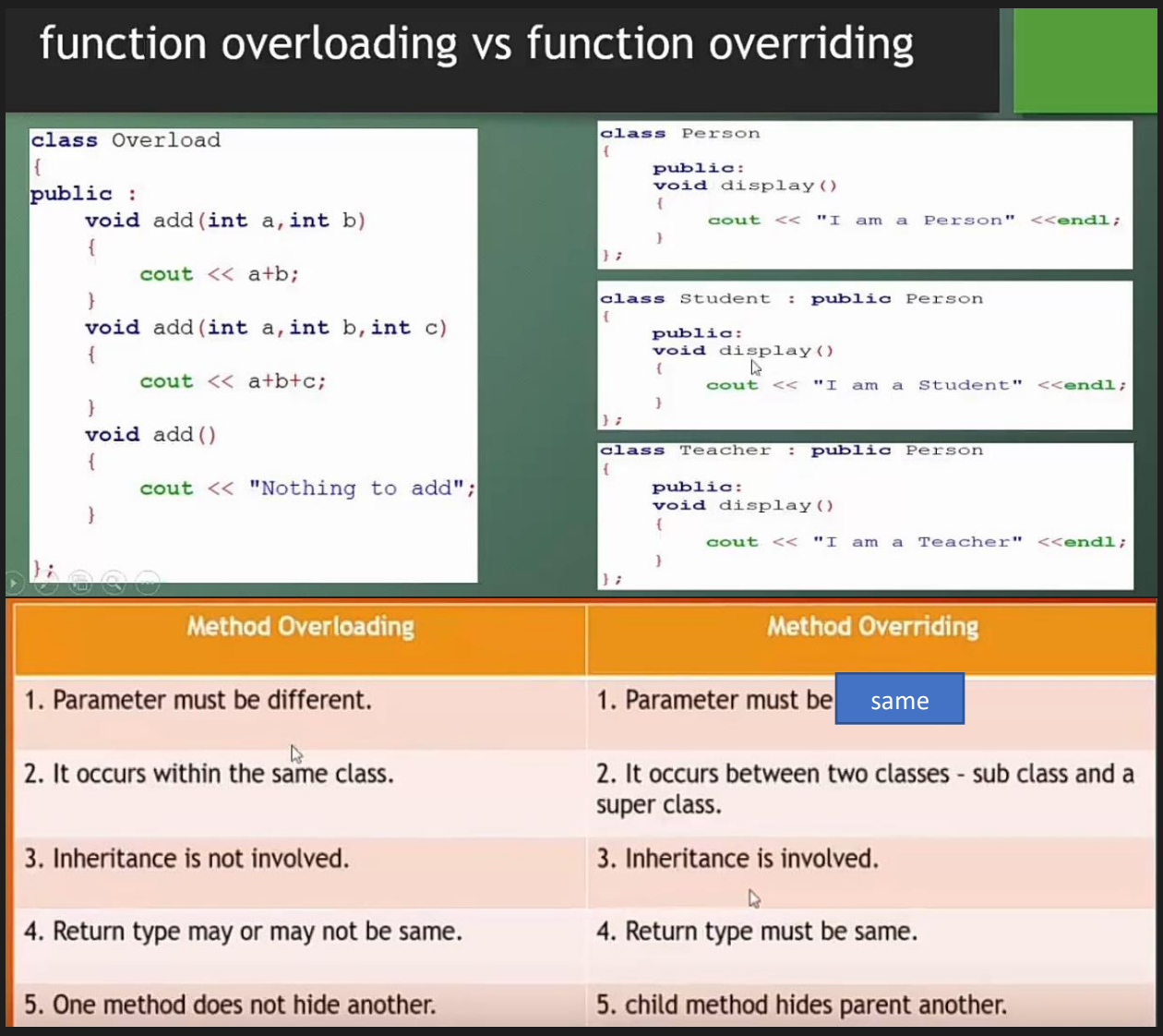
```
#include <iostream>
using namespace std;
class Person
{
public:
    void display(){
        cout << "I'm Person class" << endl;
    }
};
class Student : public Person
{
public:
     void display(){
        cout << "I'm Student class" << endl;
    }
};
```

```
class Teacher : public Person
{
public:
     void display(){
        cout << "I'm Teacher class" << endl;
    }
};
int main()
{
    Person p;
    p.display();
    Student s;
    s.display();
    Teacher t;
    t.display();
    return 0;
}
```

Output:

```
I'm Person class
I'm Student class
I'm Teacher class
```

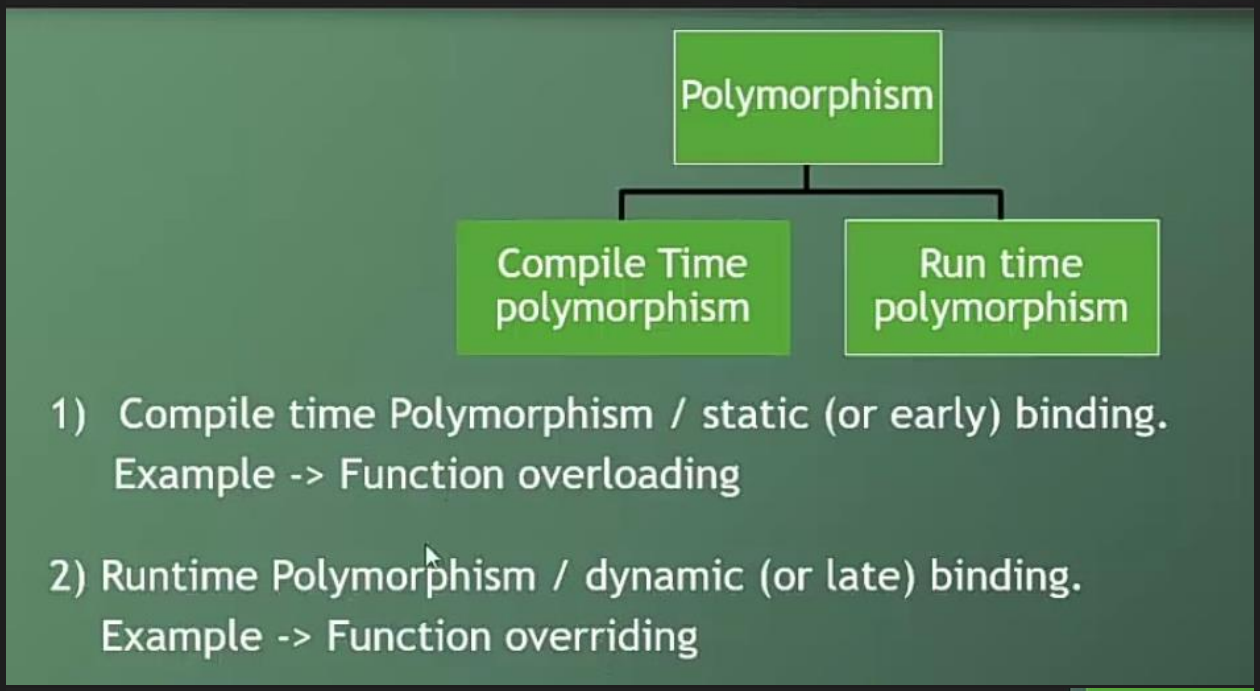
82. Overloading vs Overriding difference.



Method Overloading	Method Overriding
1. Parameter must be different.	1. Parameter must be same
2. It occurs within the same class.	2. It occurs between two classes - sub class and a super class.
3. Inheritance is not involved.	3. Inheritance is involved.
4. Return type may or may not be same.	4. Return type must be same.
5. One method does not hide another.	5. child method hides parent another.

83. Polymorphism theory and example with pointer.

Types of polymorphism



Function overloading

```
class Overload
{
public :
    void add(int a,int b)
    {
        cout << a+b;
    }
    void add(int a,int b,int c)
    {
        cout << a+b+c;
    }
    void add()
    {
        cout << "Nothing to add";
    }

};
```

```
int main()
{
    Overload obj;
    obj.add();
    obj.add(10,20);
    obj.add(10,20,30);

}
```

Function overriding

```
class Person
{
    public:
    void display()
    {
        cout << "I am a Person" <<endl;
    }
};
```

```
class Student : public Person
{
    public:
    void display()
    {
        cout << "I am a Student" <<endl;
    }
};
```

```
class Teacher : public Person
{
    public:
    void display()
    {
        cout << "I am a Teacher" <<endl;
    }
```

```
int main()
{

    Student s;
    s.display();

    Teacher t;
    t.display();

}
```

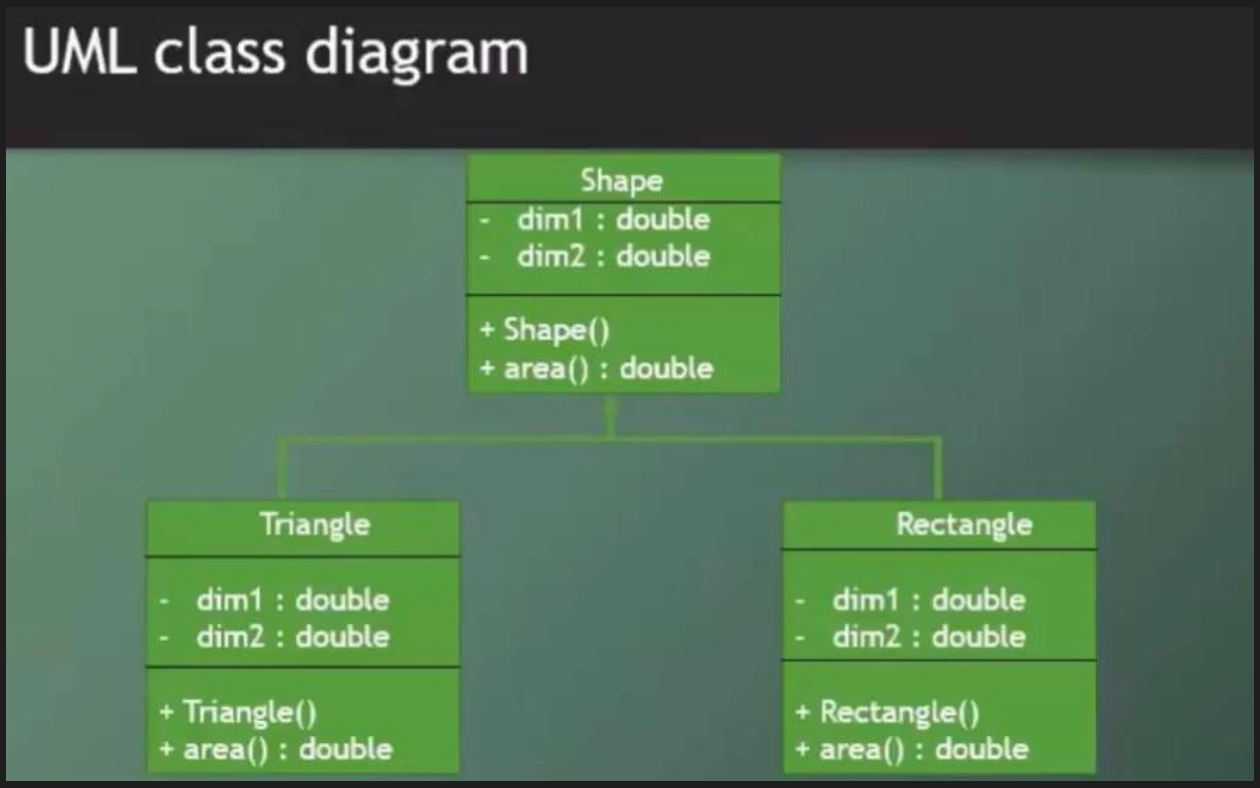
```cpp
#include <iostream>
using namespace std;
class Person
{
public:
    virtual void display(){
        cout << "I'm Person class" << endl;
    }
};
class Student : public Person
{
public:
     void display(){
        cout << "I'm Student class" << endl;
    }
};
```

```
class Teacher : public Person
{
public:
     void display(){
        cout << "I'm Teacher class" << endl;
    }
};
int main()
{
    Person *p;
    Student s;
    Teacher t;

    p = &s;
    p -> display();
    p = &t;
    p -> display();
}
```

Output:

```
I'm Student class
I'm Teacher class
```

84. UML class diagram.



```cpp
#include <iostream>
using namespace std;
class Shape
{
public:
    double dim1, dim2;
    Shape(double dim1, double dim2){
        this -> dim1 = dim1;
        this -> dim2 = dim2;
    }
    double area(){
        return 0;
    }
};
class Triangle : public Shape
{
public:
    //dim1, dim2, area duto choly aschy.
    Triangle(double dim1, double dim2)
    : Shape(dim1, dim2)
    {

    }
    double area(){
        return 0.5 * dim1 * dim2;
    }
};
```

```
class Rectangle : public Shape
{
public:
    Rectangle(double dim1, double dim2)
    : Shape(dim1, dim2)
    {

    }
    double area(){
        return dim1 * dim2;
    }
};
int main()
{
    Shape s(10, 5);
    Triangle t(10, 5);
    Rectangle r(10, 5);

    cout << "Area of triangle is : " << t.area() << endl;
    cout << "Area of rectangle is : " << r.area() << endl;
}
```

Or, (below)

```
#include <iostream>
using namespace std;
class Shape
{
public:
    double dim1, dim2;
    Shape(double dim1, double dim2){
        this -> dim1 = dim1;
        this -> dim2 = dim2;
    }
    virtual double area(){
        return 0;
    }
};
class Triangle : public Shape
{
public:
    //dim1, dim2, area duto choly aschy.
    Triangle(double dim1, double dim2)
    : Shape(dim1, dim2)
    {

    }
    double area(){
        return 0.5 * dim1 * dim2;
    }
};
```

```
class Rectangle : public Shape
{
public:
    Rectangle(double dim1, double dim2)
    : Shape(dim1, dim2)
    {

    }
    double area(){
        return dim1 * dim2;
    }
};
int main()
{
    Shape *s;
    Triangle t(10, 5);
    Rectangle r(10, 5);

    s = &t;
    cout << "Area of triangle is : " << s->area() << endl;
    s = &r;
    cout << "Area of rectangle is : " << s->area() << endl;
}
```

Output:

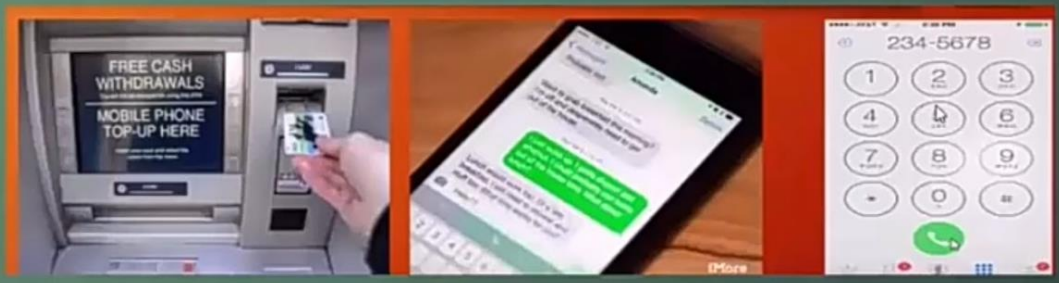
```
Area of triangle is : 25

Area of rectangle is : 50
```

85. Abstraction theory and example.

Abstraction

- What is Abstraction ?

  Abstraction is the process of hiding the implementation details and showing only the functionality to the user.



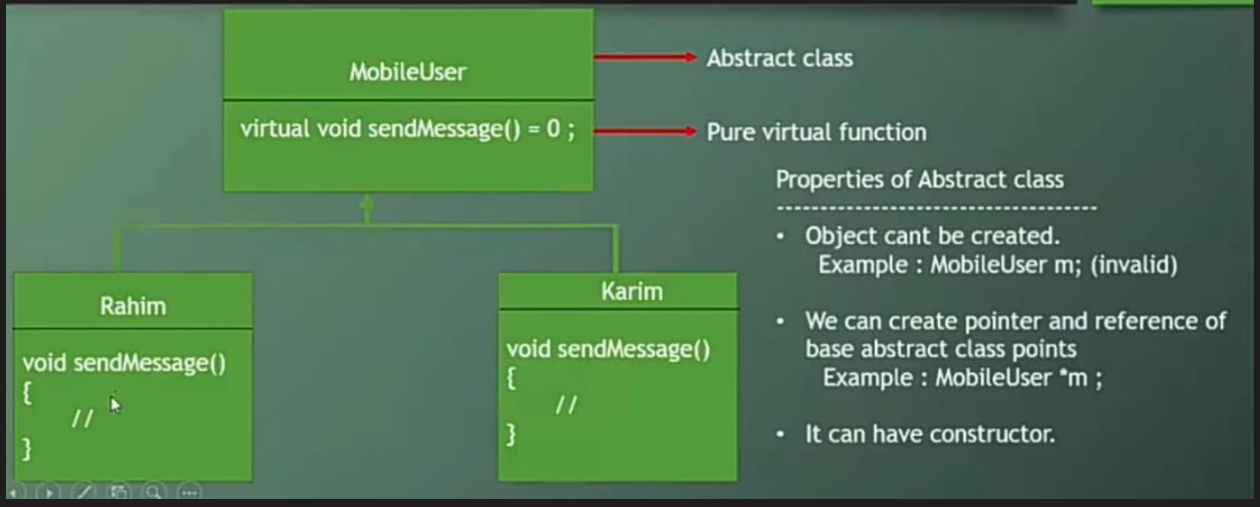
Abstraction

```
void sendMessage()
  {
    cout << "This is a message" ;
  }
```

```
virtual void sendMessage()
  {
    cout << "This is a message" ;
  }
```

```
virtual void sendMessage() = 0 ;
```

Abstraction Example



Properties of Abstract class

- Object cant be created.
  Example : MobileUser m; (invalid)
- We can create pointer and reference of base abstract class points
  Example : MobileUser *m ;
- It can have constructor.

```cpp
#include <iostream>
using namespace std;
class Mobileuser{
public:
    virtual void sendMessage() = 0;
    void display(){
        cout << "Inside the display function" << endl;
    }
};
class Rahim : public Mobileuser{
public:
    void sendMessage(){
        cout << "I'm Rahim" << endl;
    }
};
class Karim : public Mobileuser{
public:
    void sendMessage(){
        cout << "I'm Karim" << endl;
    }
};
```

```
int main()
{
    Mobileuser *m;
    Rahim r;
    Karim k;

    m = &r;
    m -> sendMessage();
    m -> display();
    m = &k;
    m -> sendMessage();
}
```

Output:

```
I'm Rahim
Inside the display function
I'm Karim
```

86. Friend class

```cpp
#include <iostream>
using namespace std;
class A{
    private:
        int id = 10;
        string name = "Kibria";
    public:
        friend class B;
};
class B{
    public:
        void display(A ob){
            cout << ob.id << endl;
            cout << ob.name << endl;
        }
};
int main(){
    A ob1;
    B ob2;
    ob2.display(ob1);
}
```

87. Exception handling theory and example.

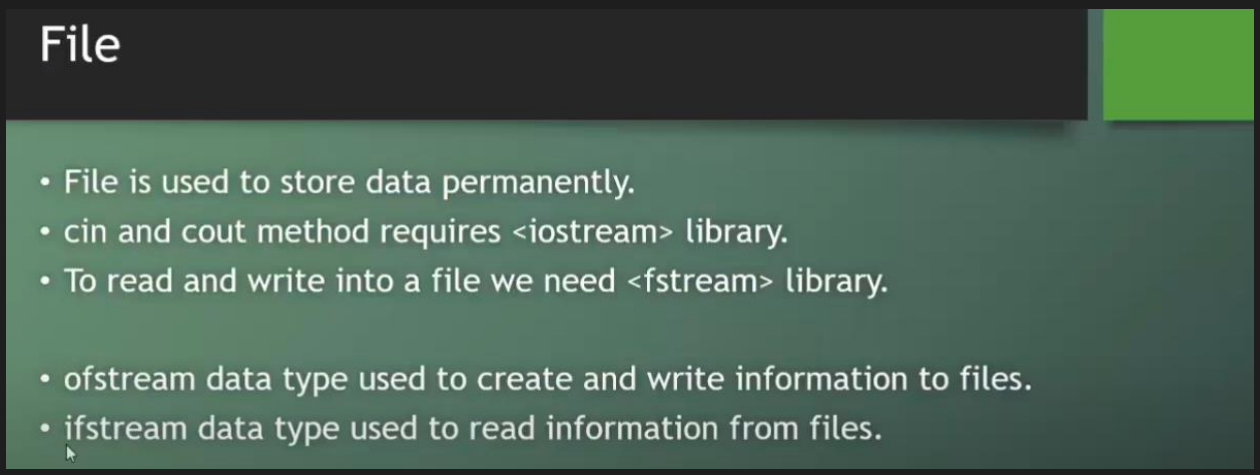
- Error -> 1) Compile Time Error, 2) Run Time Error
- Exception : Exception is a run time error.
- Exception Handling is a mechanism that can handle the exception.
- Keywords : try, catch, throw

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

int main()
{
    try
    {
        int num1, num2;
        cout << "Enter an integer: ";
        cin >> num1;
        cout << "Enter an integer: ";
        cin >> num2;
        if (num2 == 0){
            throw -1; //j kono value deya jaby.
        }
        double result = (double)num1 / num2;
        cout << "Resutl is : " << result << endl;
    }
     //catch(…) 3 ta dot dileo hoby.
    catch (int x){
        cout << "Please try again!!" << endl;
    }
}
```

Output:

```
Enter an integer: 23
Enter an integer: 2
Resutl is : 11.5

Enter an integer: 23
Enter an integer: 0
Please try again!!
```

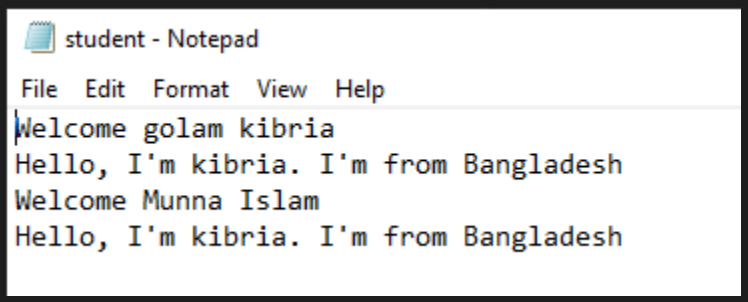
88. How to create and write into a file.



how to create and write into a file

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

int main(){
    string name;
    ofstream file;

    file.open("student.txt", ios::out|ios::app);

    cout << "Enter your name: ";
    getline(cin, name);
    file << "Welcome " << name << endl;

    file << "Hello, I'm kibria. I'm from Bangladesh\n";
    file.close();
}
```

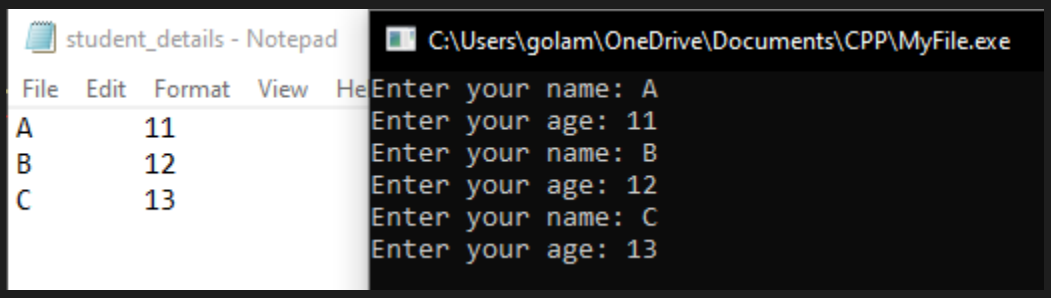
//ios::out|ios::app ata use korychi cuz file a notun kory kisu likly jaty ager ta delete na hoye jay. app dara append bujay.



89. Storeing studens details in a file.

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

int main()
{
    string name;
    int age;

    ofstream file;
    file.open("student_details.txt", ios::out|ios::app);

    for(int i = 1; i<=3; i++)
    {
        cout << "Enter your name: ";
        getline(cin, name);
        file << name << "\t";
        cout << "Enter your age: ";
        cin >> age;
        file << age << endl;
        cin.ignore();
    }
    file.close();
}
//age neyar por kono character thakly taky ignore korbo.cin.ignore()
```

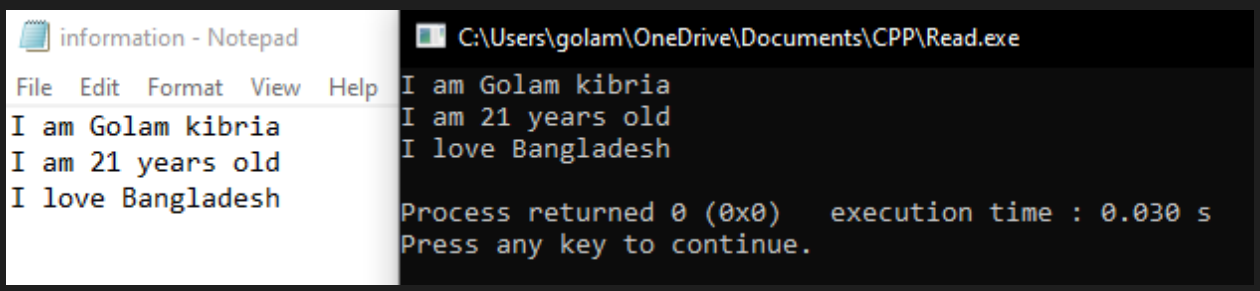
Output:


student_details - Notepad
File Edit Format View He
A 11
B 12
C 13
C:\Users\golam\OneDrive\Documents\CPP\MyFile.exe
Enter your name: A
Enter your age: 11
Enter your name: B
Enter your age: 12
Enter your name: C
Enter your age: 13

90. Read from a file

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

int main()
{
    string line;
    ifstream file("information.txt");
    while(getline(file, line)){

        cout << line << endl;
    }
    file.close();
}
```

Output:



References

- www.beginnersbook.com
- www.sololearn.com
- www.tutorialspoint.com