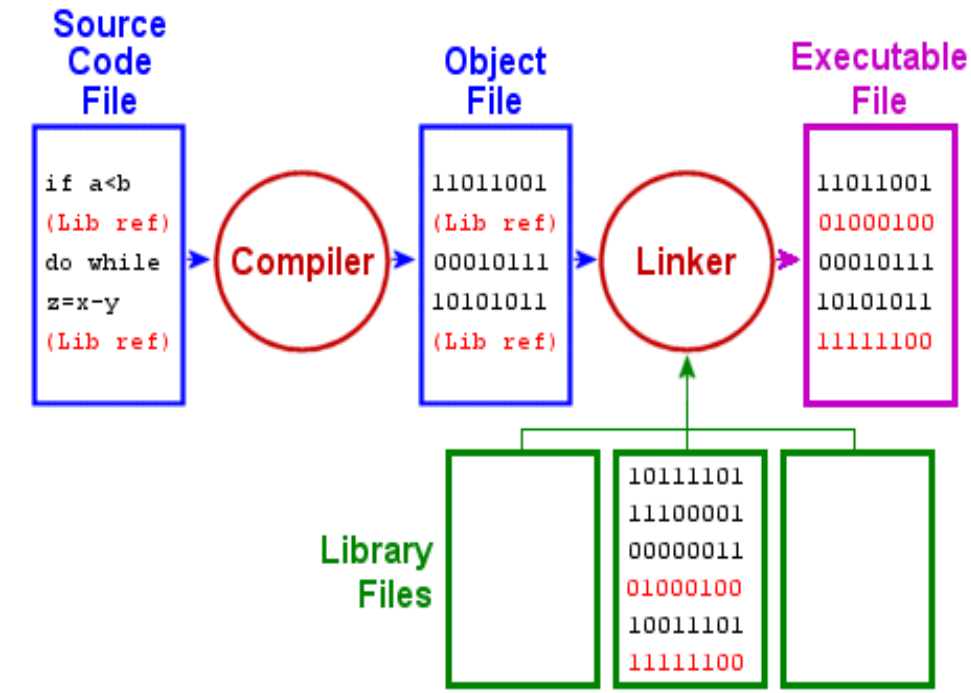
PROGRAMMING IN C++

KAUSIK DATTA
24-Oct-2017



Objectives

- Understand the operator Overloading

Operator vs Function

Operator

- Usually written in infix notation
- Examples:

```
Infix: a + b; a ? b : c;
Prefix: ++a;
Postfix: a++;
```

- Operates on one or more operands, typically up to 3 (Unary, Binary or Ternary)
- Produces one result
- Order of operations is decided by precedence and associativity
- Operators are pre-defined

Function

- Always written in prefix notation
- Examples:

```
Prefix: max(a, b);
qsort(int[], int, int,
void (*)(void*, void*));
```

- Operates on zero or more arguments
- Produces up to one result
- Order of application is decided by depth of nesting
- Functions can be defined as needed

Source: Programming in C++ by Prof. Partha Pratim Das

Operator Function in C++

- Introduces a new keyword: operator
- Every operator is associated with an operator function that defines its behavior

Operator Expression	Operator Function
a + b	Operator + (a, b)
a = b	Operator = (a, b)
c = a + b	Operator = (c, operator + (a, b))

- Operator functions are implicit for predefined operators of built-in types and cannot be redefined
- An operator function may have a signature as:

```
MyType a, b; // An enum or struct
MyType operator+(MyType, MyType); // Operator function
a + b // Calls operator+(a, b)
```

- C++ allows users to define an operator function and overload it

Source: Programming in C++ by Prof. Partha Pratim Das

Operator Overloading - Summary of Rules

- No new operator such as **, <>, or &j can be defined for overloading
- Intrinsic properties of the overloaded operator cannot be change
  - Preserves precedence
  - Preserves associativity
- These operators can be overloaded:
  + - * / % & | ~ ! = += -= *= /= %= = &= |= << >> >>= <<= == != < > <= >= && || ++ -- , ->* -> ( ) [ ]
- For unary prefix operators, use: MyType& operator++(MyType& s1)
- For unary postfix operators, use: MyType operator++(MyType& s1, int)
- The operators :: (scope resolution), . (member access), .* (member access through pointer to member), sizeof, and ?: (ternary conditional) cannot be overloaded
- The overloads of operators &&, ||, and , (comma) lose their special properties: short-circuit evaluation and sequencing
- The overload of operator-> must either return a raw pointer or return an object (by reference or by value), for which operator-> is in turn overloaded

Source: Programming in C++ by Prof. Partha Pratim Das

Operator Overloading - Example

```
#include <iostream>
using namespace std;
enum E {C0 = 0, C1 = 1, C2 = 2};
E operator + (const E& a, const E& b)
{
    unsigned int uia = a, uib = b;
    unsigned int t   = (uia + uib) % 3;
    return (E) t;
}
int main()
{
    E a = C1, b = C2;
    E x = a + b;
    cout << x << endl;
    return 0;
}
```

Source: Programming in C++ by Prof. Partha Pratim Das

Assignments

- In the previous example check the value of `a + b` without overloading + operator
- Write a class mystring like the following including constructor and destructor

```
Class mystring
{
Char *str;
}
```

- Overload operator "+" for this class to concat two strings and print the output. Do not use and C function for string like strcpy, strcat etc.
- For example:

```
mystring S1("ISI "); mystring S2("Kolkata");
mystring S3 = S1 + S2;
S3 will be "ISI Kolkata";
```

Assignments

- Develop a set of classes to store different shapes
  - Triangle
  - Square
  - Circle
- And write member function “area()” to calculate area of each type of shapes
- The classes will contain required members to store information for calculation area
- For example circle class will have a member radius