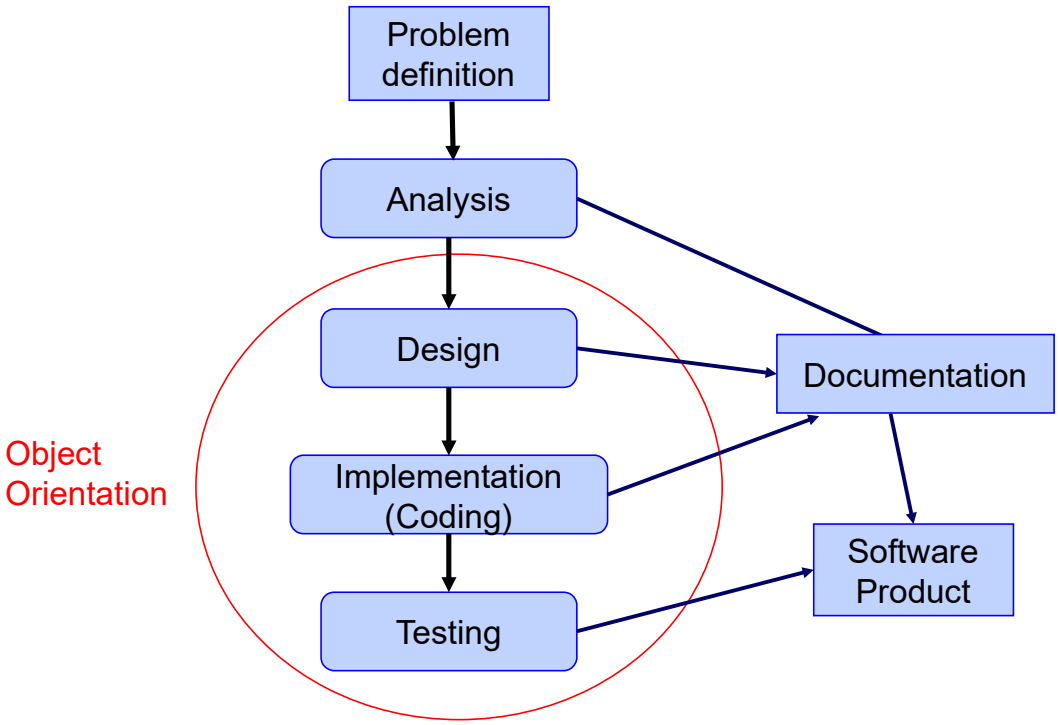
Lecture 1

Introduction

Topics

- Software Development Process
- Object Oriented Approach
- Principles of Object Oriented Programming
- Example: Graphics drawing program

Software Development Process



Software Development Process

ANALYSIS: Understanding the requirements for given problem.

DESIGN: Identifying the entities.
In object-oriented design, entities are **objects**.

CODING: Implementation in a programming language.

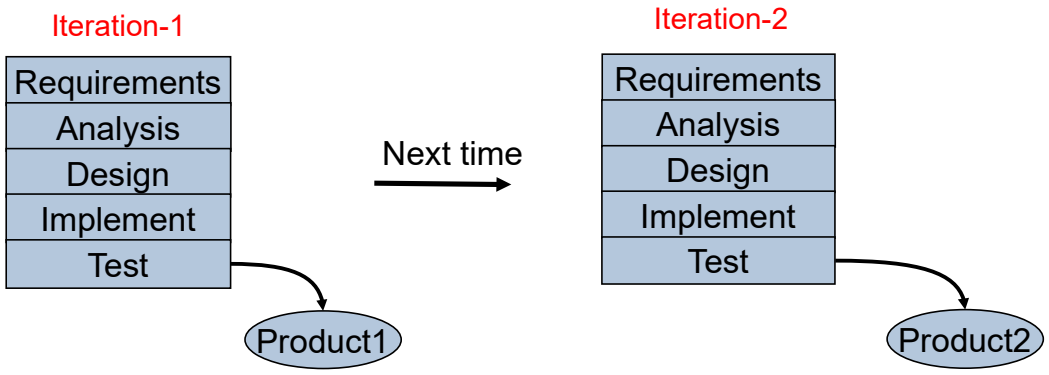
DOCUMENTATION: Writing technical reports for development team, and user manual for customers.

TESTING: The functions of each object and the whole program must be tested for possible inputs and expected outputs.

Unified Process Method in Software Development

- A software development process describes an approach to building, deploying, and maintaining a software.
- The **Unified Process** is an **iterative** software development process for building object-oriented systems.
- Development is organized into a series of mini-projects called iterations.
- Each iteration includes its own analysis, design, implementation, and testing activities.

Unified Process Method in Software Development



An iteration has a fixed time.

User view of Program features

A program must have the following features:

- Runs **correctly**.
- Runs **reliably**.
- Performs as **fast** as necessary.
- Does not waste system **resources** too much. (Processor time, Memory, Disk).
- **Easy to up-grade** the program (re-installation).
- Have sufficient **documentation** of users manuals.

Software developer view of Program features

A program must have the following features:

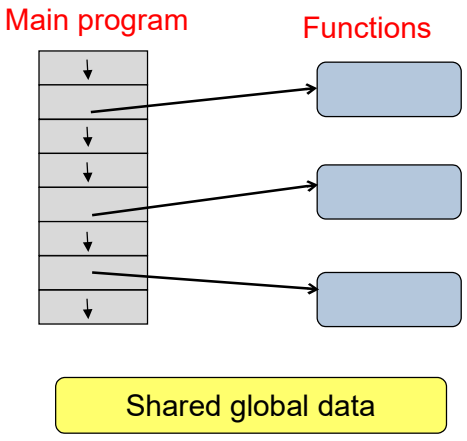
- Source code must be **readable and understandable.**
- It must be **easy to maintain and update** (change) the program, according to new requirements.
- An error should not affect other parts of a program.
- Modules of program must be **reusable** in further projects.
- It must have sufficient **design documentation.**

Programming Process

- Program development is based on **models** of real world situations.
- Computer programs are the implementions (coding) of models.
- **Modelling** is the design of a software.
- The followings are the tools for software modelling.
  - **UML diagrams** are used for design of classes.
  - **Flow charts** are used for design of algorithms.
- **Implementation** is the coding with a programming language such as C++.

Procedural Programming

- In a procedural language such as C or Fortran, the emphasis is on **functions, not objects**.
- A program is divided into **functions.**
- Main program and functions can use shared global data, as well as the passed parameters.



Disadvantages of Procedural Programming

- **Data** is less emphasized, **functions** are more emphasized.
- Procedural programs don't model the real world very well. (The real world does not just consist of functions.)
- To add new data items, all the functions that access the shared data must be **modified,** so that they can access the new items.

Topics

- Software Development Process
- Object Oriented Approach
- Principles of Object Oriented Programming
- Example: Graphics drawing program

Object-Oriented Approach

- The fundamental idea behind object-oriented programming is :
  - **The real world consists of objects.**
- Thinking in terms of objects, rather than functions, makes the software design easier.
- To solve a programming problem in an object-oriented language, the programmer asks **how the problem will be divided into objects**.
- A problem will be easier to understand and handle, if organized as **objects**.

Example1 : University System

A University System software may contain the following entities (objects):

Students have an identification number (ID) and courses attended. They take grades, their GPAs are calculated.

Instructors give courses, they perform some projects, they have some administrative duties.

Courses are given at specific times in a specific classroom. They have a plan, they have a list of enrolled students.

Example2 : Publications System

In a Publications System (Bookstore or Library) entities (objects) may be the followings:

Publishers: Contains publisher ID, publisher name, phone number, address. A publisher may publish many books.

Authors: Contains author ID, author fullname, email address. An author may have many books.

Books: Contains book ISBN number (book ID), book title, number of pages, price, its author and publisher ID keys. A book may have only one author, and only one publisher.

Building Blocks of Object Oriented Programming (OOP)

- **OOP** is a programming technique that organizes software design around data, or objects; instead of functions.
- **Classes** are programmer-defined data types that act as the blueprint for individual objects, attributes and methods.
- **Objects** are instances (variables) of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity.
- **Attributes** are datas defined in the class and represent the state of an object. Objects will have data stored in the attributes field.
- **Methods** are functions that are defined inside a class that describe the behaviors of an object. Programmers use methods for re-usability or keeping functionality encapsulated inside one object.

Topics

- Software Development Process
- Object Oriented Approach
- Principles of Object Oriented Programming
- Example: Graphics drawing program

Principles of Object Oriented Programming

- **Encapsulation :**
  - Encapsulation principle states that all important information (data) is contained inside a class and only some information is accessed.
  - The implementation and data are privately held inside a defined class.
  - Other objects do not have access to this class or the authority to make changes.
  - They are only able to call public functions (methods).
  - Encapsulation provides coding security and avoids unintended data corruption.

Principles of Object Oriented Programming

- **Inheritance :**
  - Classes can re-use code from other classes.
  - Relationships and sub-classes between objects can be assigned, enabling developers to reuse common codes while still maintaining a unique hierarchy.
  - Inheritance reduces development time and ensures a higher level of accuracy.

Principles of Object Oriented Programming

- **Polymorphism :**
  - Objects are designed to share behaviors and they can take on more than one form.
  - The program will determine which meaning or usage is necessary for each execution of that object from a base class, reducing the need to duplicate code.
  - A derived class is then created, which extends the functionality of the base class.
  - Polymorphism allows different types of objects to pass through the same interface.

Advantages of OOP

- The advantages of OOP include the followings :
  - Readability (understandability)
  - Low probability of errors
  - Easy maintenance
  - Modularity
  - Re-usability
  - Productivity
  - Scalability
  - Efficiency

- OOP works very well for complex and large projects that require continous updates and maintenance.
- Examples of such programs include operating systems, compilers, manufacturing and design.

Object Oriented Programs

- Real-world objects have two parts:
  - **Attributes** (Data)
  - **Methods** (Functions)

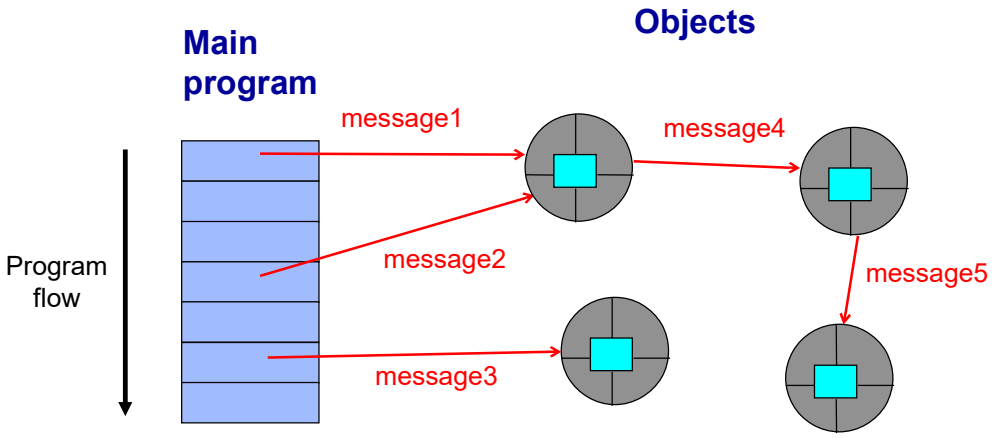
- Software objects correspond to real-world objects.
- Examples of software objects:
  - **Graphics program:** Point, Line, Rectangle, Circle, etc.
  - **Mathematics:** Complex numbers, Matrix
  - **Graphical user interface (GUI):** Windows, Menus, Buttons, Toolbars
  - **Data structures:** Arrays, Stacks, Queues, Linked Lists

Object Oriented Programs

- To create software models of real world objects, both *data* and *functions* are combined into a single program entity (Class).
- In OOP, data and its functions are **encapsulated** into a single entity (class).
- A C++ class is a structure declaration similar to a C struct.
- An object is an instance (variable) of a specific class.
- The data of an object can be private, so it cannot be accessed directly.
- The private data can only be changed through its functions (also known as its public interface).
- Classes simplify writing, debugging, and maintaining the program.

Structure of an Object Oriented program

- In OOP, objects combine member data and member functions.
- A C++ program consists of a number of objects that communicate with each other by calling member functions.
- **Messages** are member **function callings** of an object with necessary parameter values.



Topics

- Software Development Process
- Object Oriented Approach
- Principles of Object Oriented Programming
- Example: Graphics drawing program

Example : Point class in a Graphics program

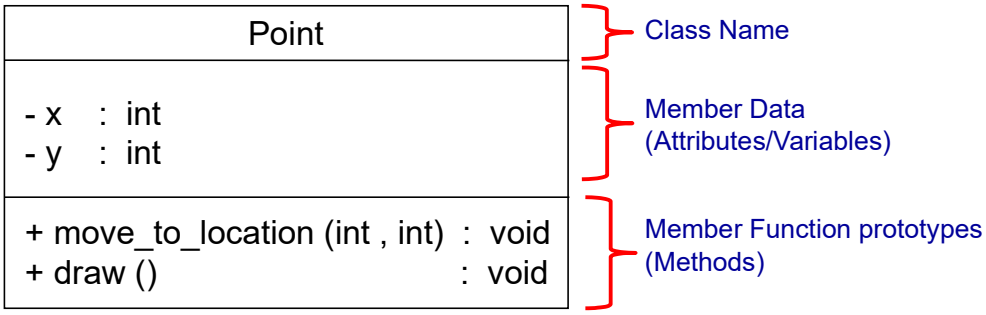
- Suppose a graphics program will read mouse clicks and movements for drawing in a graphics window.
  (Similar to the Brush utility in Microsoft Windows Paint Program.)
  - User left-clicks the mouse to start drawing.
  - User moves the mouse to do drawing.
  - User right-clicks the mouse to finish drawing.

The Point class can be defined with following members.

- Integer variables **(x, y)** : Coordinates of a point.
- **move_to_location()** function : Moves a point to a new (x, y) coordinate.
- **draw()** function : Draws a line from previous point to current mouse point on graphics screen.

UML class diagram for the Point class

- An **UML (Unified Modeling Language)** class diagram is used as a design tool for modelling of a class. It has three sections.
  - Top section : Contains name of class.
  - Middle section : Contains declarations of member variables (data) of class.
  - Bottom section : Contains prototypes of member functions of class.

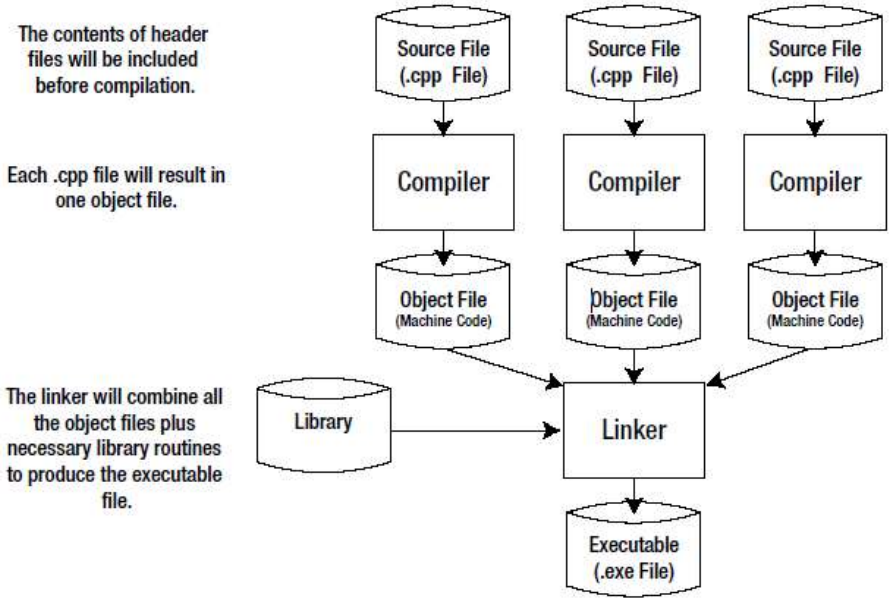


(Access specifier symbols: - is private, + is public)

Borland Graphics Library

- Standard C++ language does not have a built-in (ready-made) graphics program library.
- Therefore, a separate graphics library should be used such as OpenGL (Open Graphics Library), or Borland, etc.
- In the example program below, the **Borland Graphics Interface** library functions are used.
- BGI library consists of two files: winbgi.cpp and graphics.h
- The project will contain the following three files:
  - **winbgi.cpp** : Library functions file
  - **graphics.h** : Header file
  - **main.cpp** : Main program file

Steps of Program Compiling and Linking process



Description of Main Program

- When the drawing program is executed, two windows are created automatically.
  - **Console Window** : Can be used for normal program input/output functions such as printf, scanf, etc.
  - **Graphics Window** : Can be used only for graphics-related functions such as initgraph, closegraph, setcolor, settextstyle, outtextxy, line, mousedown, etc.

main.cpp File

```
#include "graphics.h"

//Graphics window dimensions
#define WIDTH 640
#define HEIGHT 480

//(continued)
```

Declaration of the Point class

The declaration of Point class should be written outside of main program.

main.cpp File

```
class Point
{
   int x, y;   // Location coordinates of a point

   public: // Access mode for member functions
   void move_to_location (int xn, int yn)
   {
        //Set the new location coordinates of point object
        x = xn;
        y = yn;
   }

   void draw ()
   {
        line(x, y, mousecurrentx(), mousecurrenty());
        //Draws a line between old Point location and new mouse location
   }

}; //end of class
```


Main program

Objects (such as the P variable below) of the Point class can be defined inside the main program.

main.cpp File

```
int main()
{
    int GraphDriver = 0, GraphMode = 0;
    // Start the graphics window
    initgraph(&GraphDriver, &GraphMode, "", WIDTH, HEIGHT);
    setbkcolor(CYAN); // Set background color
    cleardevice();    // Clear screen with background color

    settextstyle(SANSSERIF_FONT, HORIZ_DIR, 2); // Set font type and size
    setbkcolor(BLACK);   // Set background color
    setcolor(GREEN);     // Set color of  texts
    outtextxy(5,5, "MOUSE");
    outtextxy(5,25, "Left click  : Start drawing");
    outtextxy(5,50, "Right click : End drawing");

    setcolor(BLUE);      // et color of drawing pen

    Point  P; // Definition of a Point class object
    bool STATUS = false;
    // Boolean flag variable (used to  enable/disable drawing)
```

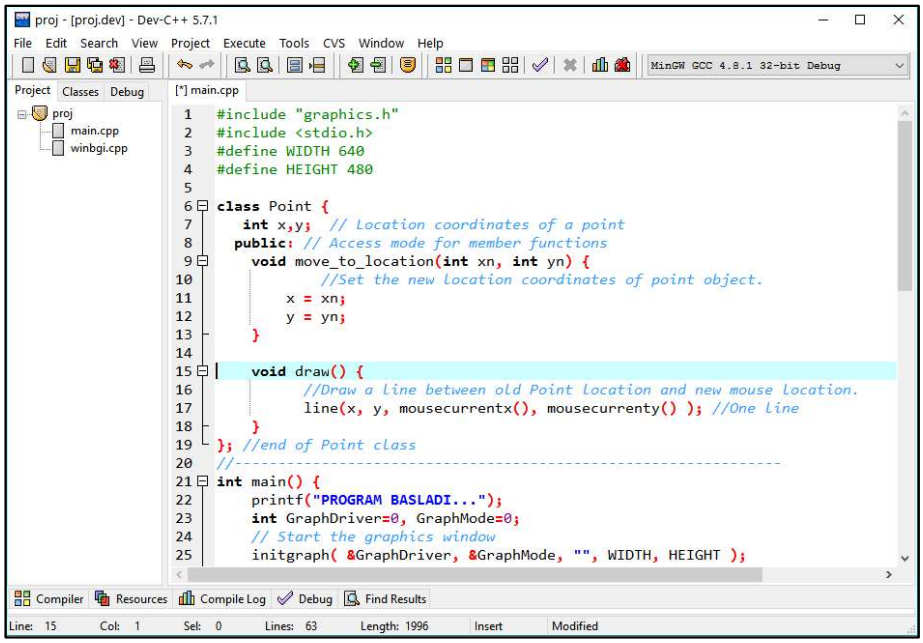
Main program (continued)

main.cpp File

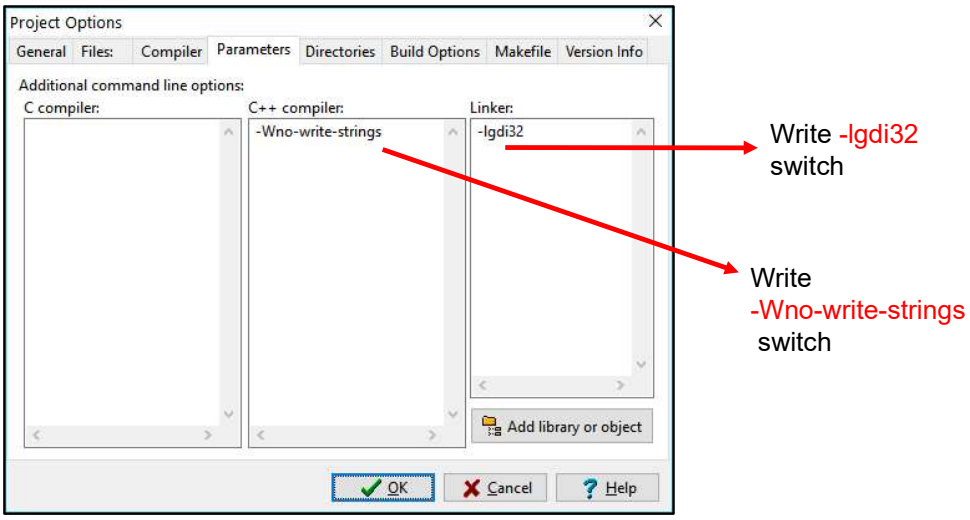
```
while (true) // Endless loop
{
  if ( mousedown() == true )  {
     if ( whichmousebutton() == LEFT_BUTTON )
     {
        STATUS = true; // Drawing is now enabled
        P.move_to_location ( mouseclickx(), mouseclicky() );
        // Set new location for object
      }
      else
        STATUS = false; // Drawing is now disabled
  }

  if (STATUS == true)
  {
     P.draw (); // Call drawing function of object
     P.move_to_location ( mousecurrentx(), mousecurrenty() );
     // Set new location for object
  }

} //end of while

} //end of main
```


Using Dev-C++ IDE for Compiling and Linking

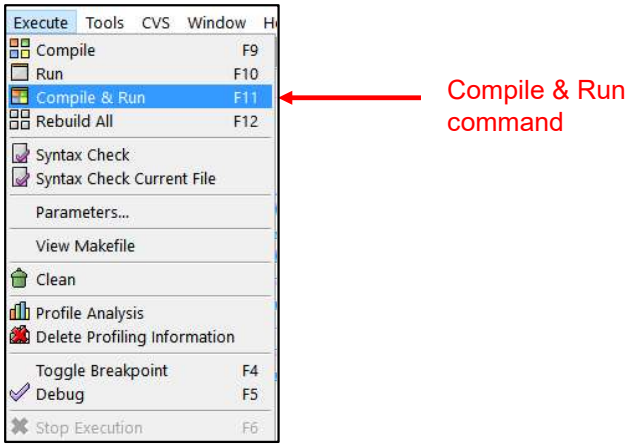
Dev-C++ is an Integrated Development Environment (IDE) that contains Text editor, Compiler, Linker, and Debugger.



Step1) From main menu select the File command, and open the project file (proj.dev).
Step2) Write the following switch options.
Go to "Project" menu and choose "Project Options".
Go to "Parameters" tab.
In "Linker" field, enter "-lgdi32"



Step3) From main menu select the Compile & Run command.



Alternative Method: Command-line for Compiling and Linking

- Write the program source file (main.cpp) with any text editor such as Notepad or Notepad++.
- Then, open a command-line console window.
- The command below calls the GNU C++ compiler.
- g++ compiles the C++ source code and generates the executable file.
- It can be used in Windows or Linux operating systems.

Command-line console window

```
g++  -Wno-write-strings  main.cpp  winbgi.cpp  -lgdi32  -o  main.exe
```

Switch disables warning strings

Switch tells to use graphics library

Switch generates object code