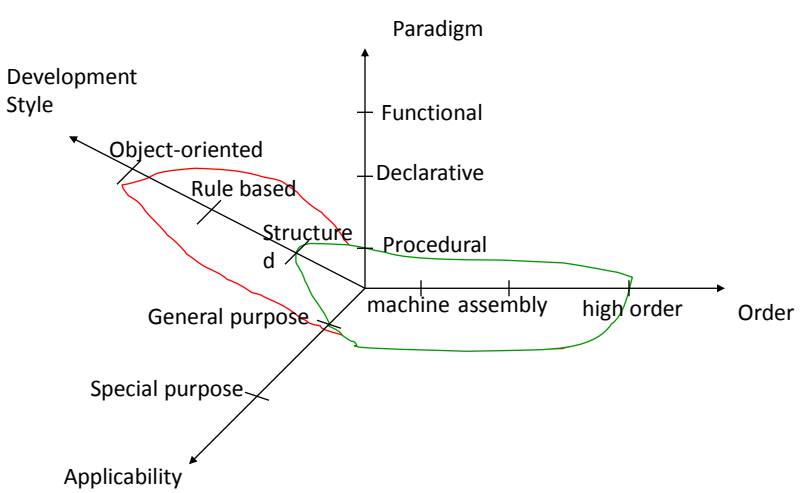
Overview of OOP and C++

Types of Programming Languages



Procedural and Object-Oriented Programming

- Procedural programming
  - focus is on the process.
  - Procedures/functions are written to process data.
- Object-Oriented programming
  - focus is on objects, which contain data and means to manipulate the data.
  - Messages sent to objects to perform operations.

Object-oriented programming

- The **class** is one of the defining ideas of **object-oriented programming**.
  - A **class** is a template definition of the method s and variable s in a particular kind of **object** .
- An **object** is a specific instance of a **class.**
  - it contains real values instead of variables.

Properties of Procedural Languages

Property 1: uncontrolled access to global variables

Difficulty
(a) -Uncontrolled modification

int x;

(b) Poor maintainability and enhancement.

Look everywhere: any function could have used it
main
input
addend
remove

Properties of Procedural Languages

Property 2: Lack of specification: cannot specify permitted operations

Difficulty: Any operation can be performed

int defines structure + permitted operations

struct: define whatever is possible

Property 3: Low modularity

Difficulty: Common data coupling by global data

Property 4: Poorly identifiable components

Difficulty: No reuse

Properties of Procedural Languages

Property 5: Strong typing

Difficulty: No reuse

Program for sorting integers cannot sort real numbers

Benefits of Object Orientation

I) No accidental modification of data

Data is within an object

2) High specification level

Associate semantically meaningful operations
'compute salary', 'print payroll' for Employee

3) Easier maintenance

Changes within an object do not affect external objects

Benefits of Object Orientation

4) High level of modularity

Only data coupling or stamp coupling

5) Reuse

- Object: a well defined unit of data and operations
- Reduce program development time by reusing objects

C++ History

- C developed by Dennis Ritchie at AT&T Bell Labs in the 1970s.
  - Used to maintain UNIX systems
  - Many commercial applications written in C
- C++ developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s.
  - Overcame several shortcomings of C
  - Incorporated object oriented programming
  - C remains a subset of C++

Background

- C++ was developed by Bjarne Stroustrup at Bell Laboratories
  - Originally called "C with classes"
  - The name C++ is based on C's increment operator (++)
    - Indicating that C++ is an enhanced version of C
- Widely used in many applications and fields
- Well-suited to "Programming in the Large"

11

What Is C++?

- (Mostly) an extension of *C* to include:–
  - Classes
  - Templates
  - Inheritance and Multiple Inheritance
  - Function and Operator Overloading
  - New (and better) Standard Library
  - References and Reference Parameters
  - Default Arguments
  - Inline Functions
  - …
- A few small syntactic differences from *C*

12

Keywords Shared with *C*

C++ keywords

Keywords common to the C and C++ programming languages

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

13

New Keywords in *C++*

C++ keywords

C++-only keywords

and	and_eq	asm	bitand	bitor
bool	catch	class	compl	const_cast
delete	dynamic_cast	explicit	export	false
friend	inline	mutable	namespace	new
not	not_eq	operator	or	or_eq
private	protected	public	reinterpret_cast	static_cast
template	this	throw	true	try
typeid	typename	using	virtual	wchar_t
xor	xor_eq			

14

A Simple C++ Program

- **<iostream>**
  - Must be included for any program that outputs data to the screen or inputs data from the keyboard using C++ style stream input/output.
  - Replaces **<stdio.h>** of C
- C++ requires you to specify the return type, possibly **void**, for all functions.
- **iostream is a library containing definitions of cin and cout**

15

A Sample C++ Program

- A simple C++ program begins this way

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

int main()
{
```

- ends this way

```
        return 0;
}
```

Explanation of code (1/5)

- Variable declaration line

  int number_of_pods, peas_per_pod, total_peas;

  – int means that the variables represent integers

Explanation of code (2/5)

- Program statement

  cout << “Press return after entering a number.\n”;

  – cout (see-out) used for output to the monitor

  – “<<“ inserts “Press…a number.\n” in the data bound for the monitor

  – Think of cout as a name for the monitor
    - “<<“ points to where the data is to end up

  – ‘\n’ causes a new line to be started on the monitor

Explanation of code (3/5)

- Program statement

  cin >> number_of_pods;

  – cin (see-in) used for input from the keyboard

  – “>>” extracts data from the keyboard

  – Think of cin as a name for the keyboard
    - “>>” points from the keyboard to a variable where the data is stored

Explanation of code (4/5)

- Program statement

  total_peas = number_of_pods * peas_per_pod;

  – Performs a computation

Explanation of code (5/5)

- Program statement

```
cout << number_of_pods;
```

Sends the value of variable number_of_pods to the monitor

Classes in O-O Programming

- Set of objects which have the same structure and display the same behaviour
  - Every class has a distinct name
    - Point, Account, Employee, Book
- A template that defines the structure and behaviour
  - Has two parts -an interface and an implementation
  - Interface -behaviour
  - Implementation - representation of properties,methods

Class itself does not respond to the operations defined in it.

Object

- Object is a representation of some information
  - Name
  - Values or properties
    - Data members
  - Ability to react to requests (messages)
    - Member functions
- When an object receives a message, one of two actions are performed
  - Object is directed to perform an action
  - Object changes one of its properties

A First Program - Greeting.cpp

```
// Program: Display greetings
// Author(s): Raj Programmer
// Date: 1/24/2001
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Hello world!" << endl;
    return 0;
}
```

Preprocessor directives

Comments

Provides simple access

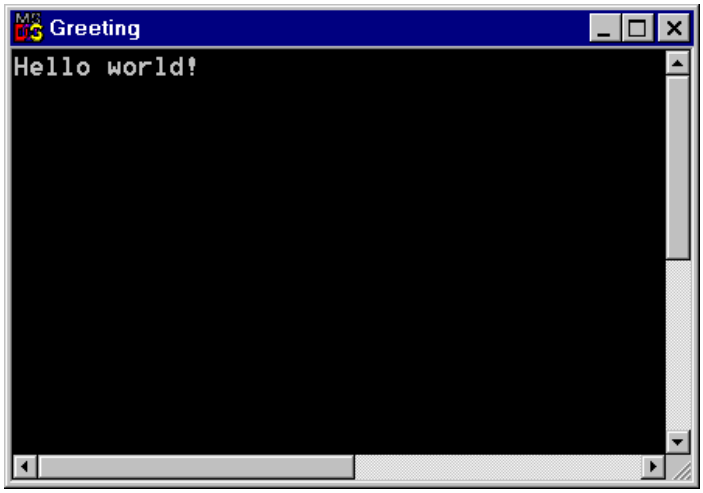
Function named main() indicates start of program

Ends executions of main() which ends program

Insertion statement

Function

Greeting Output



Area.cpp

```
#include <iostream>
using namespace std;
int main() {
   // Extract length and width
   cout << "Rectangle dimensions: ";
   float Length;
   float Width;
   cin >> Length >> Width;

   // Compute and insert the area

   float Area = Length * Width;

  cout << "Area = " << Area << " = Length "
    << Length << " * Width " << Width <<
  endl;
   return 0;
}
```

Comments

- C++ has two conventions for comments
  - // single line comment (preferred)
  - /* long comment */ (save for debugging)

Compiling *C++*

- Use **gcc**, Visual Studio, etc.
- File types

  .cc, .cp, .cpp, .CPP, .cxx, .c++, .C

  .h, .H

Some of these have special properties.

28

A Simple C++ Example

```
// C++ simple example

#include <iostream>  //for C++ Input and Output
int main ()
{
  int number3;

  std::cout << "Enter a number:";
  std::cin >> number3;

  int number2, sum;

  std::cout << "Enter another number:";
  std::cin >> number2;

  sum = number2 + number3;
  std::cout << "Sum is: " << sum <<std::endl;

  return 0;
}
```

C++ style comments

standard output stream object
stream insertion operator

stream extraction operator
standard input stream object

stream manipulator
Concatenating insertion operators

29

Notes on Simple C++ Program

- Stream manipulator **`std::endl`**
  - Outputs a newline.
  - Flushes the output buffer

Note: **`std::ends`** flushes the buffer but does not add *newline*.

- The notation **`std::cout`** specifies a name (`cout`) that belongs to the *namespace* **`std`**.

Namespace: a generalization of *scope*.

C++ allows access to multiple namespaces with the ' :: ' operator