Queens

College of Arts and Science for Women
Punalkulam, Near Thanjavur, Gandarvakkottai,
Pudukkottai (Dt).

Mrs.A. Ramya.,

Assistant Professor,

Department of BCA,

Subject Name : Programming in C++

Subject Code : 16SCCCS2/ 16SCCCA2

Unit : 5

Topic : Manipulating Strings ,

Object Oriented system

MANIPULATING STRINGS

Introduction:

- A string is a sequence of characters.
- For using the string class <string> will be included in the header file.
- Using constructor, member functions and operators,the following operations will be executed.

(i) Creating string objects.
(ii) Reading string objects from keyboard.
(iii) Displaying string objects to the screen.
(iv) Finding a substring from a string.
(v) Modifying string objects.
(vi) Comparing string objects.
(vii) Adding string objects.
(viii) Accessing characters in a string.
(ix) Obtaining the size of strings.

String constructor:

Constructor	Usage
String();	For creating an empty string.
String (const chat * str);	For creating a string object from a num terminated string.
String(const string &str);	For creating a string object from other string object.

String function:

Function		**Task**
1) append( )	-	Appends a part of string to another string.
2) assign()	-	Assigns a partial string.
3) at()	-	Obtains the character stored at a specified location.
4) compare()	-	Compares two strings.
5) empty( )	-	Returns true ,if the string is empty.
6) erase()	-	Removes characters.
7) find( )	-	Search for the occurrence of a specified location.
8) insert( )	-	Inserts characters in a specified location.
9) length( )	-	Gives the number of elements in a string.
10) size( )	-	Gives the number of characters in the string.
11) swap( )	-	Swaps the given string with the invoking string.
12) begin( )	-	Returns the reference to the start of the string.
13) end( )	-	Returns the reference to the end of the string.

Operators for string objects:

Operator	Meaning
=	Assignment
+	Concatenation
+=	Concatenation assignment
<	Less than
<=	Less than or equal
>=	Greater than or equal
>	Greater than
[ ]	Subscription
<<	Output
>>	Input

Creating string objects:

- User can create string objects in number of ways.
- Examples are,

```
string s1;              // Using constructor with on arguments.

string s2("xyz");    // Using one  argument constructor.

  s1=s2;                  // Assigning

cin>>s1;                // Reads one word from keyword

getline(cin,s1);       // Read one line.
```

Manipulating string objects:

- To modify contents of string objects using the member functions such as insert(),replase(),erase() and append().

syntax:

```
insert(location,strobj);

erase(location,No.of.char);

replace(location,No.of.charstrobj);
```

Source program:

```
    #include<iostream.h>

    #include<string.h>

int main()

    {

        string s1("12345");

        string s2("abcde");

        cout<<"original strings are:";

        cout<<"s1:"<<s1<<"\n";

        cout<<"s2":<<s2<<"\n";

        cout<<"Place s2inside s1\n");

        s1.insert(4,s2);

        cout<<"modified s1:"<<s1<<"\n";

        cout<<Remove 5 characters from s1\n";

        s1.erase(4,5);
        cout<<"Now s1:"<<s1<<"\n";
        cout<<"Replace charecters";
        s2.replace(1,s,s1);
        cout<<"Now s2:"<<s2<<"\n";
        cout<<"Now s2:"<<s2<<"\n";
        return 0;
    }
```

OUTPUT:

```
                Orginal strings are:
                 s1:12345
                 s2:abcde
                 place s2 inside s1
                 modified s1:1234abcde5
                 Remove 5 character from s1
                 Now s1:12345
                 Replace middle characters
                 Now s2:a12345e
```

ACCESSING CHARACTERS IN STRINGS:

- To access substrings and individual character of a string in several ways.
- The string class supports the following function

  at( ) - For accessing individual characters.

  sub str( ) - For retrieving a substring.

find( ) - For finding a specified substring

find_first_of( ) - For finding the location of first occurrence of the specified characters

find_last_of() - For find the location of last occurrence of the specified characters.

SOURCE PROGRAM:

```
#include<iostream.h>
#include<string.h>
int main()
{
    string s("ONE TWO THREE FOUR");
    cout<<"the string contains:";
    for(inti=0;i<$.length();i++)
    cout<<s.at(i);
    int x1=s.find("two");
    cout<<"\n\n TWO is found at:"<<x1;
    int x2=s.find_first_of('T');
    cout<<"\n t is found first at:"<<x2;
    int x3=s.find_last_of('R');
    cout<<"R is found at last:"<<x2;
    cout<<"\n Retrieve and print substring two";
    cout<<s.substr(x1,3)
```

```
   return 0;
  }
```

OUTPUT:

```
  The string contains:
     ONE TWO THREE FOUR
     Two is found at:4
     T is found first at:4
     R is found at last:17
   Retrive and print substring two
     TWO.
```

COMPARING AND SWAPPING:

*)The compare() function can be used to combare either two string or portions of two strings.

*)The swap() function can be used for swapping the contents of two string objects.

SOURCE PROGRAM:

```
#include<iostream.h>
#include<string.h>
int main()
{
  string s1("Read");
  string s2("Read");
  cout<<"s1="<<s1<<"\n";
cout<<"s2="<<s2<<"\n";
int x=s1.compare(s2);
          if(x==0)
cout<<"s1==s2"<<"\n";
          else if(x>0)
cout<<"s1<s2"<<"\n";
cout<<"\n Before swap \n";
cout<<"s1="<<s1<<"\n";
cout<<"s2="<<s2<<"\n;
           s1.swap(s2);
cout<<"\n After swap\n";
cout<<"s1="<<s1<<"\n";
cout<<"s2="<<s2<<"\n";
           return 0;
          }
```

OUTPUT:

```
    s1=Road
    s2=Read
    s1>s2
    Before swap
     s1=Road
      s2=Read
```

After swap

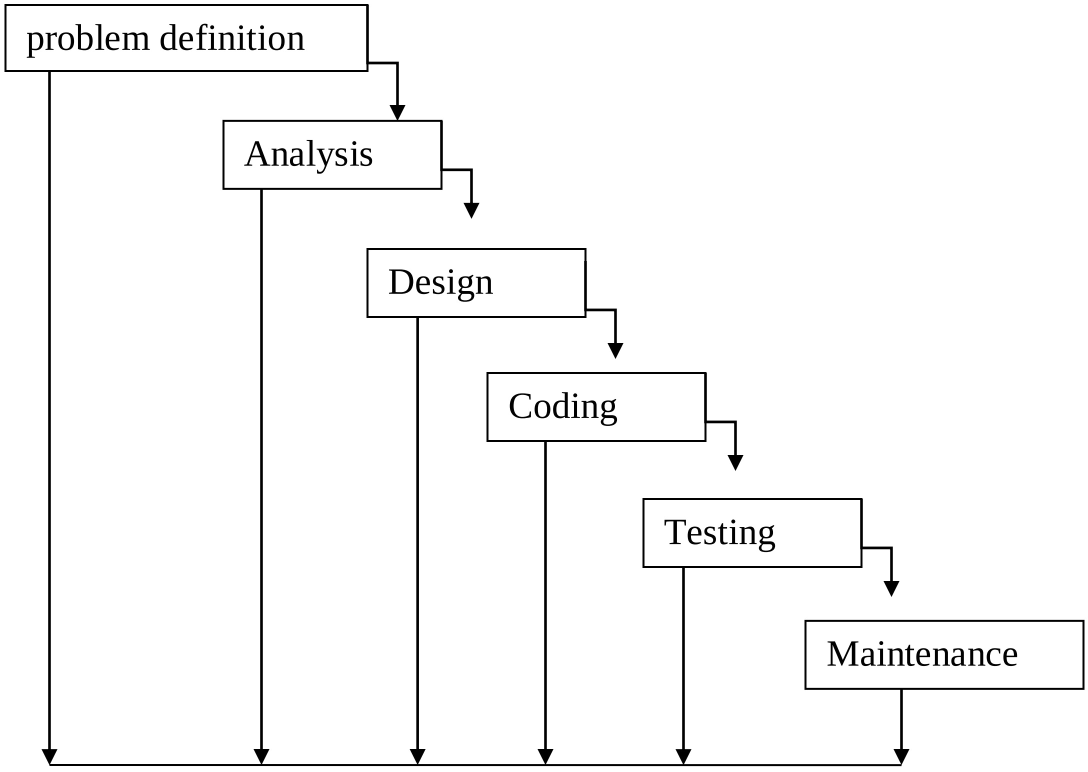
s1=Read

s2=Road

OBJECT-ORIENTED SYSTEM DEVELOPMENT

Procedure-Oriented Paradigms

- Software development is characterized by a series of stages depicting the various tasks.
- The classic software life cycle is used for the procedure oriented development.
- The life cycle is referred to as the "Water fall" model.



It contains the following stages

1) PROBLEM DEFINITION:

- The activity requires a precise definition of the problem in user terms.
- A clear statement of the problem.
- This stage answer the question 'why'.

2)ANALYSIS:

- This cover a details study of requirement.
- This stage answer the question 'what'.
- What are the input to the system?
- What are the process required?
- What are the outputs expected?
- What are the constraints?

3)DESIGNS:

- It deals with various concepts of system design such as data structure software , software architecture and algorithms.
- This stage answers the question "how".

4)CODING:

- Coding refers to the translation of the design into machine readable form.
- This stage answer the question "how".

5)TESTING:

- One the code is return, it should be tested for correctness.
- This stage answers the questions "What, when and how".

6)MAINTENANCE:

- After the software has been installed it may face some changes.
- Maintenance ensure that these changes are incorporated whenever necessary.

Output of software life cycle:

Phase	Output
Problem Definition (why)	1)Problem statement sheet. 2)Project request.
Analysis (what)	1)Requirements documents. 2)Specification documents.
Design(how)	1)Design document. 2)Test class design.
Coding(how)	1)Code document. 2)Test plan.
Testing(what and how)	1)Tested code. 2)Tested results.
Maintenance	1)Maintenance log sheets. 2)Version document.

- Software life cycle is implemented using the functional decomposition technique also known as top-down approach or modular approach.

Software Development Tool (OR)

Procedure - Oriented Development Tools:

PROCESS	FIRST GENERATION	SECOND GENERATION	THIRD GENERATION
Physical process	System flow charts	Context diagram	Inheritance,graphs,object relationship charts
Data representation	Layout form grid charts	Data dictionary	Object dictionary
Logical process	Play script English narrative	Decision tables tree	Data flow diagrams
Program representation	Program flow chart	Structure charts	State change diagrams

Software Development Tools Are,

1)System flowcharts.

2)Program flowcharts.

3)Play scripts.

4)Layout forms.

5)Grid charts.

6)Context diagrams.

7)Data flow diagrams.

8)Data dictionary.

9)Structure chart.

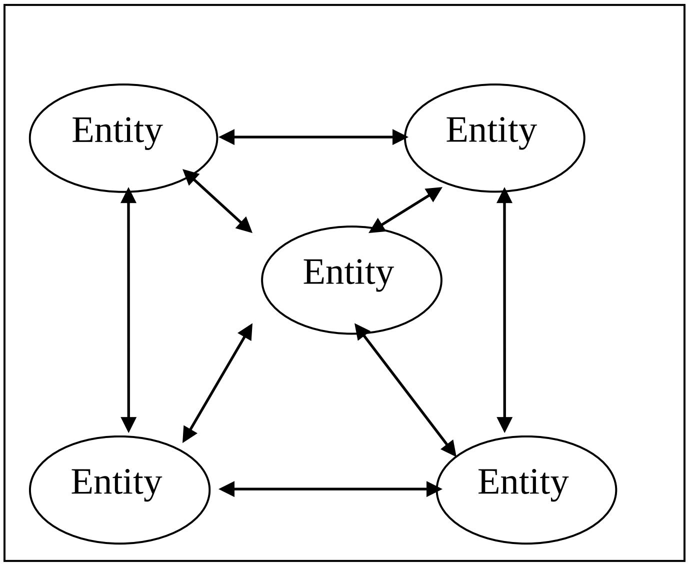
10)Decision table.

11)Decision tree.

OBJECT ORIENTED PARADIGM:

- One object oriented paradigm draws on the general system.
- A system can be viewed as a collection of entities,that interact together to accomplish certain objectives.
- Entities may represent physical object such as equipment and people abstract concepts such as data files and functions.
- In object oriented analysis the entities are called object.

RELATIONSHIP OF ENTITIES:



- As the name indicates, the object oriented paradigm emphasis on the object that encapsulation data and procedures.

Fountain model of object oriented software development:

- Object - oriented analysis (OOA) refers to the method of specifying requirement.
- Object-oriented design (OOD) turns the software requirement, into specification for objects.
- Objects oriented programming (OOP) refers to the implementation of the program using object.

Object oriented notations and graphs:

- Graphical notation are an essential part of any design and development process.
- Need notations to represent classes, objects subclasses, and their inter relationship.
- Following notations are commonly used:

1)Classes- and objects.

2)Instances of objects.

3)Message communication between objects.

4)Inheritence relationship.

5)Classification relationship.

6)Composition relationship.

7)Hierarchical relationship.

8)Client-server relationship.

9)Process layering.

Steps in object oriented analysis:

- Understanding the problem.
- Drawing the specifications of requirement of the user and the software.
- Identifying the objects and their attributes.
- Identifying the services that each object is excepted to provide.
- Establishing interconnections between the objects in terms of services required and services rendered.

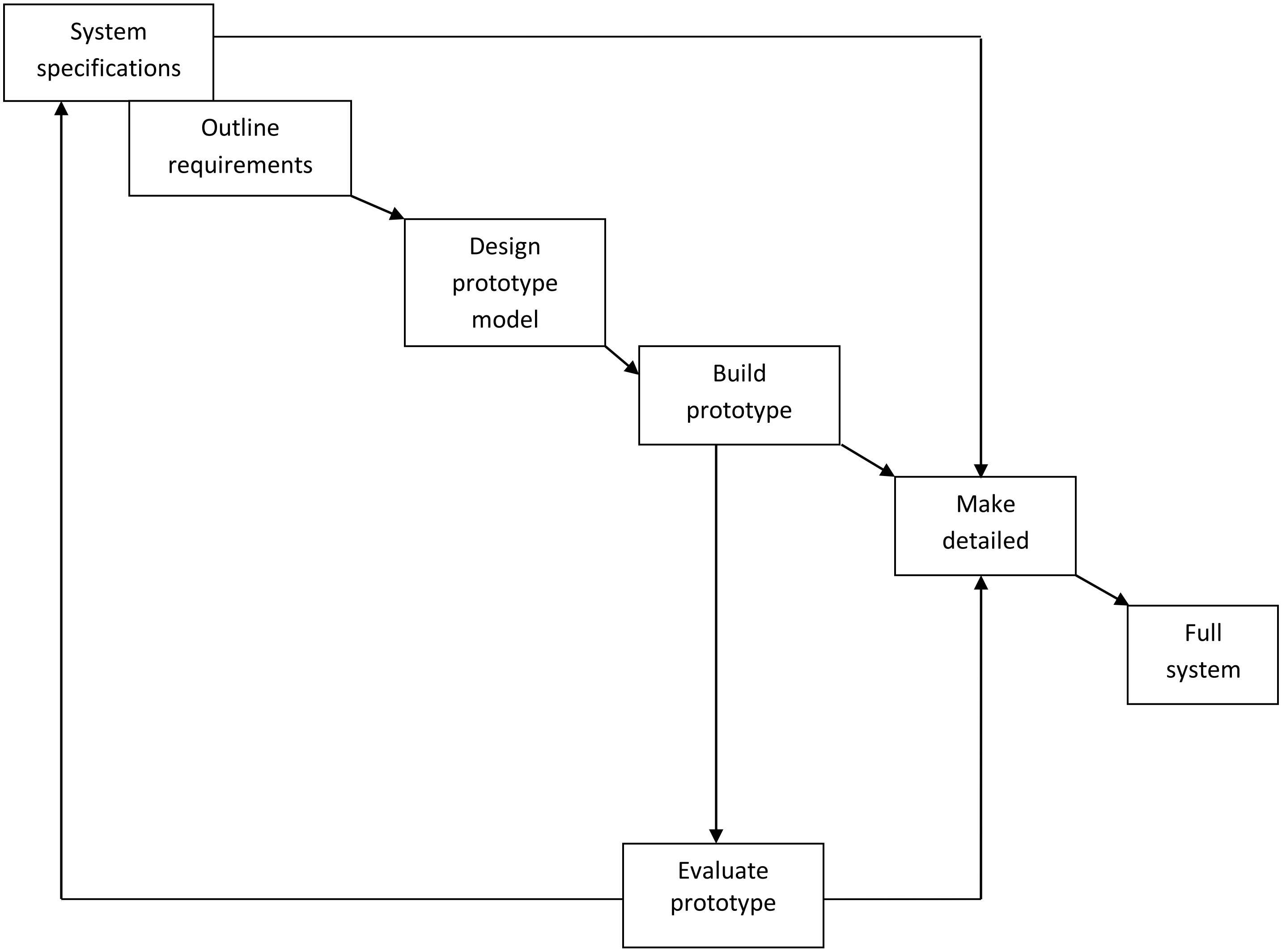
Steps in object oriented design:

- Review of objects created in the analysis phase.
- Specification of class dependencies.
- Organization of class hierarchies.
- Design of classes.
- Design of member functions.
- Design of driver program.

Prototyping paradigm:

- The real world application problems are complex and structure of the system the precise requirements at the beginning.
- A prototype is a scaled down version of the system and source requirements.
- Developer and customer agree upon "outline specification" of the system.

- The prototype is evaluated and built.
- Produce understandable specifications which are correct and complete as possible.
- The user can understand what is being offered.
- Maintenance changes that are required when a system is installed and minimized.
- Development engineers can work from a set of specifications which have been tested and approved.


System specifications
Outline requirements
Design prototype model
Build prototype
Make detailed
Full system
Evaluate prototype