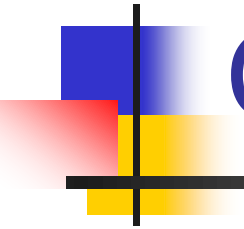
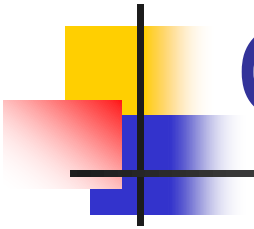
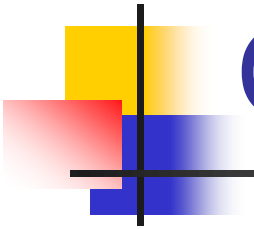


Coding and Unit Testing



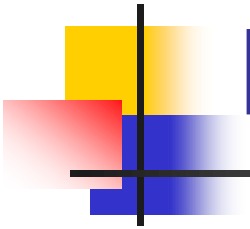
Coding

- Goal is to implement the design in best possible manner
- Coding affects testing and maintenance
- As testing and maintenance costs are high, aim of coding activity should be to write code that reduces them
- Hence, goal should not be to reduce coding cost, but testing and maint cost, i.e. make the job of tester and maintainer easier



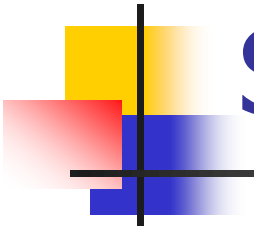
Coding…

- Code is read a lot more
  - Coders themselves read the code many times for debugging, extending etc
  - Maintainers spend a lot of effort reading and understanding code
  - Other developers read code when they add to existing code
- Hence, code should be written so it is easy to understand and read, not easy to write!



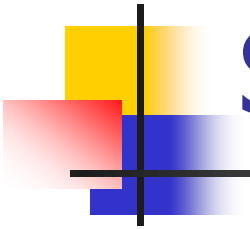
Programming Principles

- The main goal of the programmer is write simple and easy to read programs with few bugs in it
- Of course, the programmer has to develop it quickly to keep productivity high
- There are various programming principles that can help write code that is easier to understand (and test…)



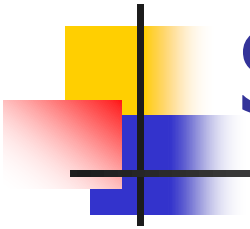
Structured Programming

- Structured programming started in the 70s, primarily against indiscriminate use of control constructs like gotos
- Goal was to simplify program structure so it is easier to argue about programs
- Is now well established and followed



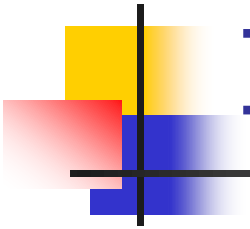
Structured Programming…

- A program has a static structure which is the ordering of stmts in the code – and this is a linear ordering
- A program also has dynamic structure –order in which stmts are executed
- Both dynamic and static structures are ordering of statements.



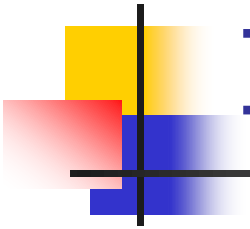
Structured Programming…

- Goal of structured programming is to write programs whose dynamic structure is same as static
- I.e. stmts are executed in the same order in which they are present in code
- As stmts organized linearly, the objective is to develop programs whose control flow is linear



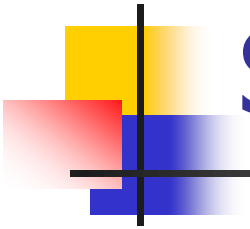
Information Hiding

- Software solutions always contain data structures that hold information
- Programs work on these DS to perform the functions they want
- In general only some operations are performed on the information, i.e. the data is manipulated in a few ways only
- E.g. on a bank's ledger, only debit, credit, check cur balance etc are done



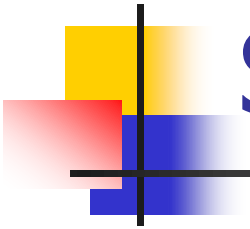
Information Hiding…

- Information hiding – the information should be hidden; only operations on it should be exposed
- I.e. data structures are hidden behind the access functions, which can be used by programs
- Info hiding reduces coupling
- This practice is a key foundation of OO and components, and is also widely used today



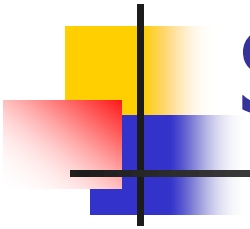
Some Programming Practices

- Control constructs: Use only a few structured constructs (rather than using a large no of constructs)
- Goto: Use them in infrequent manner, and only when the alternatives are worse
- Info hiding: Use info hiding
- Use-defined types: use these to make the programs easier to read



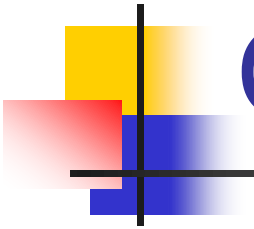
Some Programming Practices..

- Nesting: Avoid heavy nesting of if-then-else; if disjoint nesting can be avoided
- Module size: Should not be too large – generally means low cohesion
- Module interface: make it simple
- Robustness: Handle exceptional situations
- Side effects: Avoid them, document



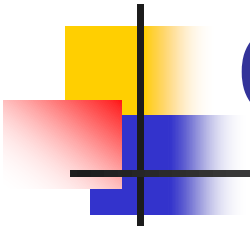
Some Programming Practices..

- Empty catch block: always have some default action rather than empty
- Empty if, while: bad practice
- Read return: should be checked for robustness
- Return from finally: should not return from finally
- Correlated parameters: Should check for compatibility



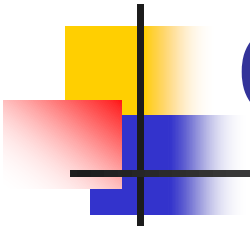
Coding Standards

- Programmers spend more time reading code than writing code
- They read their own code as well as other programmers code
- Readability is enhanced if some coding conventions are followed by all
- Coding standards provide these guidelines for programmers
- Generally are regarding naming, file organization, statements/declarations, …
- Some Java conventions discussed here



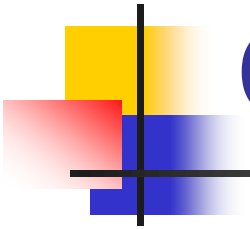
Coding Standards…

- Naming conventions
  - Package name should be in lower case (mypackage, edu.iitk.maths)
  - Type names should be nouns and start with uppercase (Day, DateOfBirth,…)
  - Var names should be nouns in lowercase; vars with large scope should have long names; loop iterators should be i, j, k…
  - Const names should be all caps
  - Method names should be verbs starting with lower case (eg getValue())
  - Prefix *is* should be used for boolean methods



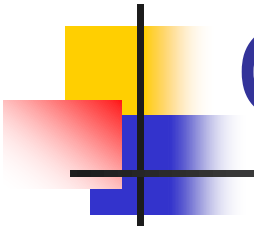
Coding Standards…

- Files
  - Source files should have .java extension
  - Each file should contain one outer class and the name should be same as file
  - Line length should be less than 80; if longer continue on another line…



Coding Standards…

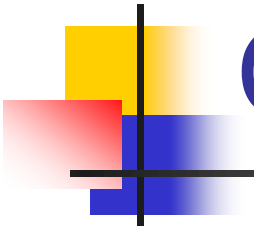
- Statements
  - Vars should be initialized where declared in the smallest possible scope
  - Declare related vars together; unrelated vars should be declared separately
  - Class vars should never be declared public
  - Loop vars should be initialized just before the loop
  - Avoid using break and continue in loops
  - Avoid executable stmts in conditionals
  - Avoid using the do… while construct



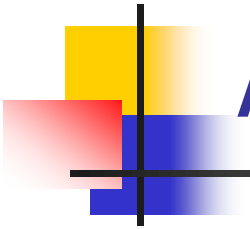
Coding Standards…

- Commenting and layout
  - Single line comments for a block should be aligned with the code block
  - There should be comments for all major vars explaining what they represent
  - A comment block should start with a line with just /* and end with a line with */
  - Trailing comments after stmts should be short and on the same line

Incrementally Developing Code

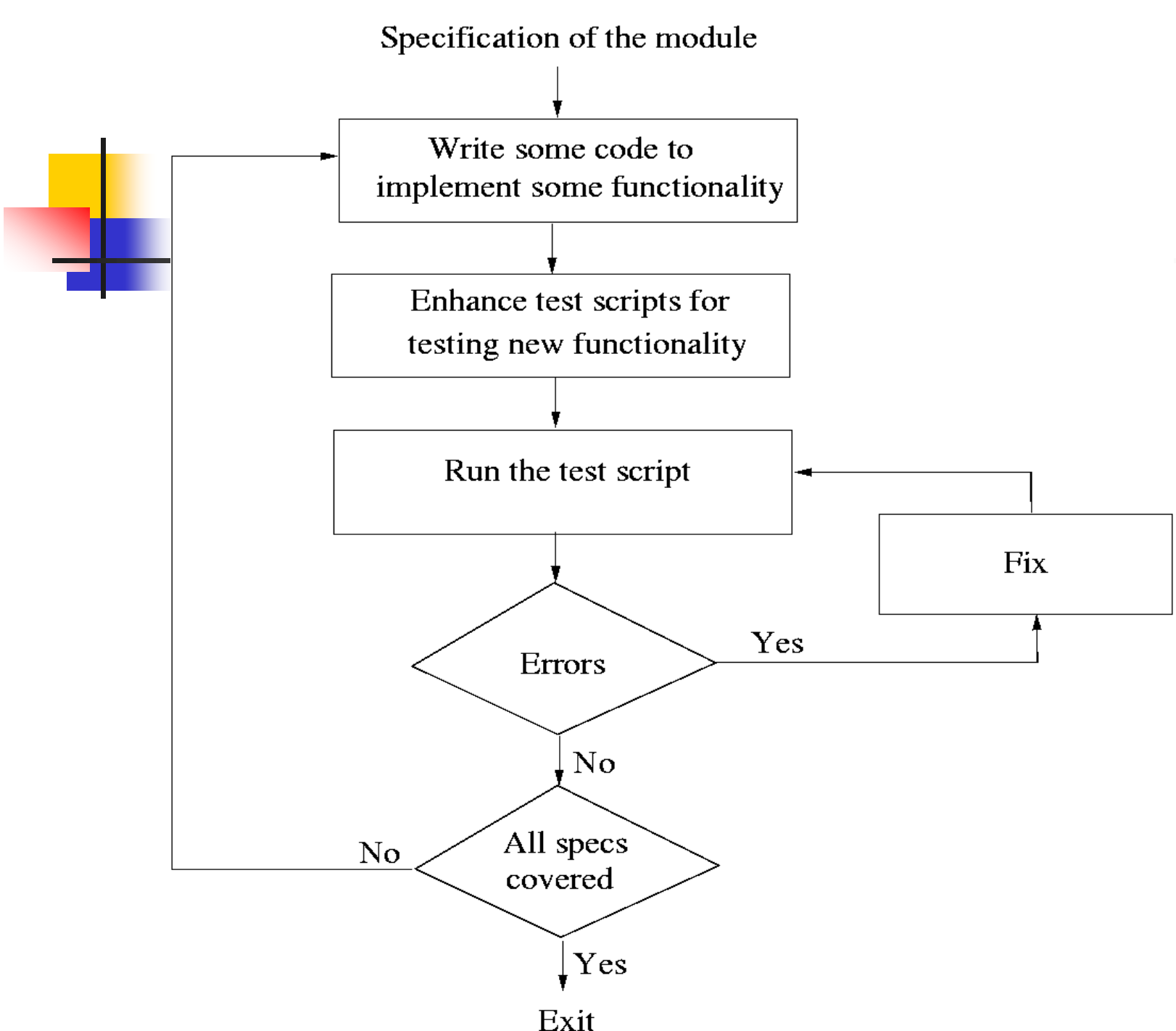


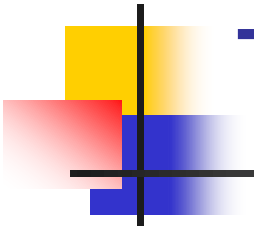
- Coding starts when specs for modules from design is available
- Usually modules are assigned to programmers for coding
- In top-down development, top level modules are developed first; in bottom-up lower levels modules
- For coding, developers use different processes; we discuss some here



An Incremental Coding Process

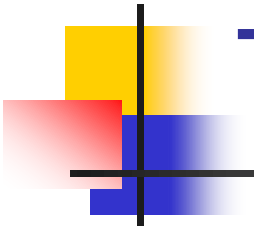
- Basic process: Write code for the module, unit test it, fix the bugs
- It is better to do this incrementally – write code for part of functionality, then test it and fix it, then proceed
- I.e. code is built code for a module incrementally


Specification of the module
Write some code to implement some functionality
Enhance test scripts for testing new functionality
Run the test script
Fix
Errors
Yes
No
All specs covered
No
Yes
Exit



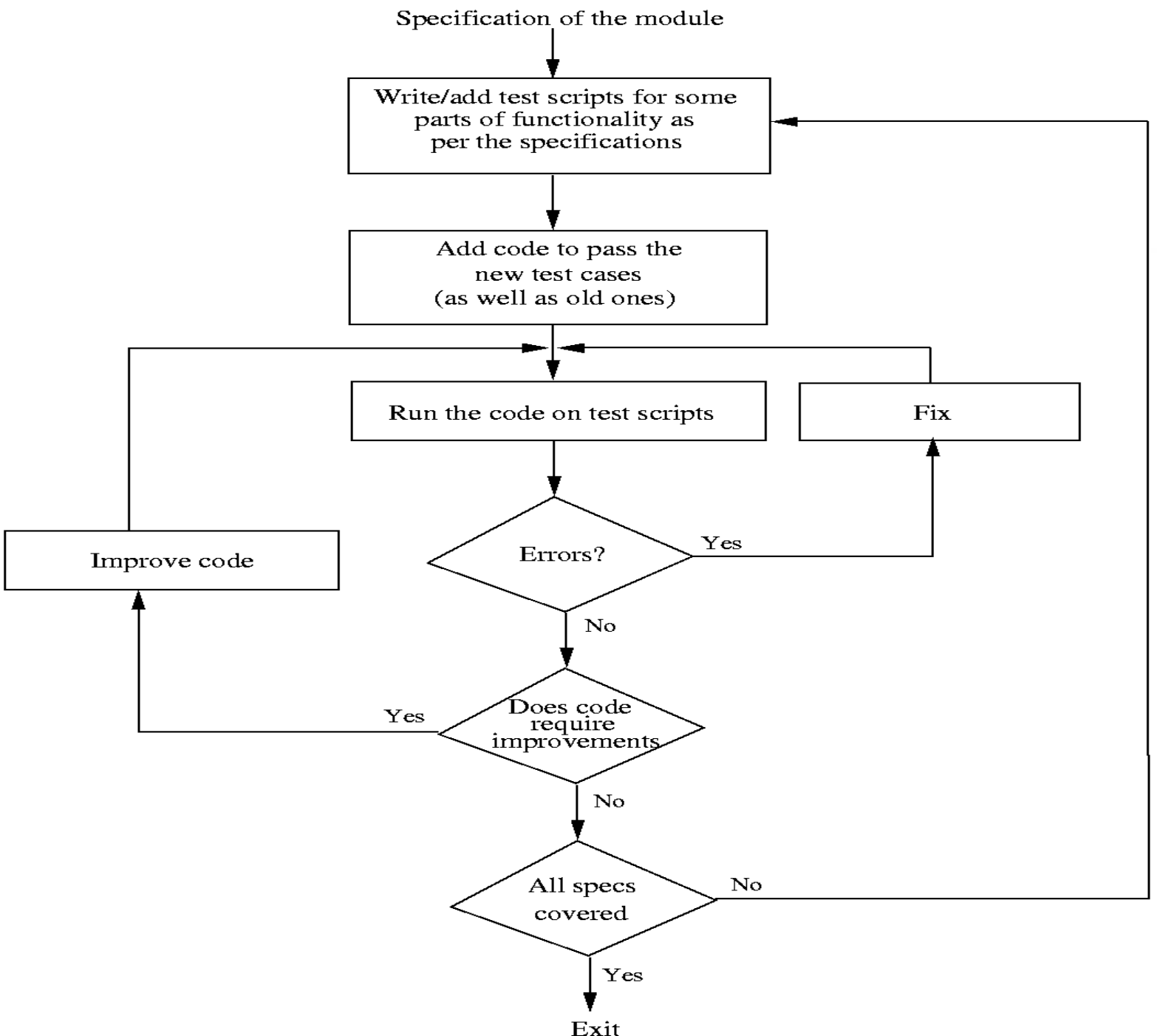
Test Driven Development

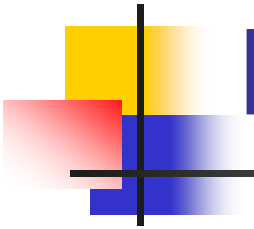
- This coding process changes the order of activities in coding
- In TDD, programmer first writes the test scripts and then writes the code to pass the test cases in the script
- This is done incrementally
- Is a relatively new approach, and is a part of the extreme programming (XP)



TDD…

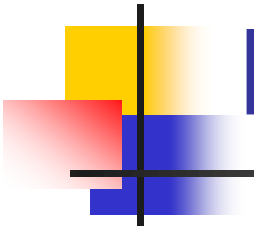
- In TDD, you write just enough code to pass the test
- I.e. code is always in sync with the tests and gets tested by the test cases
  - Not true in code first approach, as test cases may only test part of functionality
- Responsibility to ensure that all functionality is there is on test case design, not coding
- Help ensure that all code is testable


Specification of the module
Write/add test scripts for some parts of functionality as per the specifications
Add code to pass the new test cases (as well as old ones)
Run the code on test scripts
Fix
Errors?
Yes
No
Improve code
Does code require improvements
Yes
No
All specs covered
No
Yes
Exit



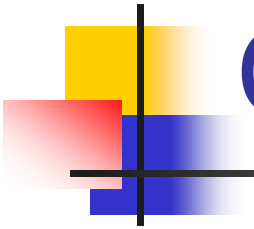
Pair Programming

- Also a coding process that has been proposed as key practice in XP
- Code is written by pair of programmers rather than individuals
  - The pair together design algorithms, data structures, strategies, etc.
  - One person types the code, the other actively reviews what is being typed
  - Errors are pointed out and together solutions are formulated
  - Roles are reversed periodically



Pair Programming…

- PP has continuous code review, and reviews are known to be effective
- Better designs of algos/DS/logic/…
- Special conditions are likely to be dealt with better and not forgotten
- It may, however, result in loss of productivity
- Ownership and accountability issues are also there
- Effectiveness is not yet fully known



Common Coding Errors

- Goal of programmer is to write quality code with few bugs in it
- Much of effort in developing software goes in identifying and removing bugs
- Common bugs which occur during coding directly or indirectly manifest themselves to a larger damage to the running program
- List of common coding errors can help a programmer avoid them

Undeclared Variables

- int main()
- {
- cin>>x;
- cout<<x;
- }
- *"Huh? Why do I get an error?"*

  Your compiler doesn't know what x means. You need to declare it as a variable.
- int main()
- {
- int x;
- cin>>x;
- cout<<x;
- }

Uninitialized variables

- int count;
- while(count<100)
- {
- cout<<count;
- count++;
- }
- Remember to initialize your variables

Setting a variable to an uninitialized value

- int a, b;
- int sum=a+b;
- cout<<"Enter two numbers to add: ";
- cin>>a;
- cin>>b;
- cout<<"The sum is: "<<sum;

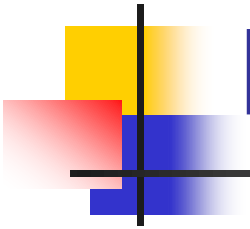
- To fix this error, move the addition step after the input line.
- int a, b;
- int sum;
- cout<<"Enter two numbers to add: ";
- cin>>b;
- cin>>a;
- sum=a+b;
- cout<<"The sum is: "<<sum;

Using a single equal sign to check equality

- char x='Y';
- while(x='Y')
- {
- //...
- cout<<"Continue? (Y/N)";
- cin>>x;
- }

Undeclared Functions

```
int main()
{
  menu();
}
void menu()
{
  //...
}
```



Extra Semicolons

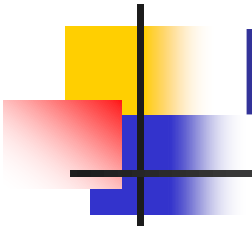
- int x;
- for(x=0; x<100; x++);
- cout<<x;

Overstepping array boundaries

- int array[10];
- //…
- for(int x=1; x<=10; x++)
- cout<<array[x];

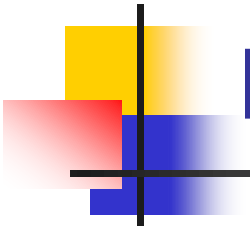
Misusing the && and || operators

- int value;
- do
- {
- //...
- value=10;
- }while(!(value==10) || !(value==20))



Memory Leaks

- A memory leak is a situation, where the memory is allocated to the program which is not freed subsequently
- Occurs frequently in the languages which do not have automatic garbage collection
- Can cause increasing usage of memory which at some point of time can lead to exceptional halt of the program



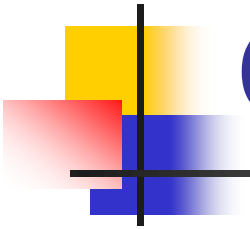
Freeing an Already Freed Resource

- Programmer tries to free the already freed resource
- May be serious, if some malloc between the two free stmts as the freed location may get allocated to a new variable, and subsequent free will deallocate the new variable.

Synchronization Errors

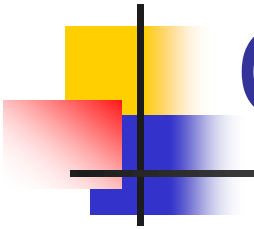
- Possible when there are multiple threads which are accessing some common resources, in a parallel program
- three categories of synchronization errors: deadlocks, race condition, live lock
- Deadlock example:

```
Thread 1:
synchronized (A){
synchronized (B){ }
}
Thread 2:
synchronized (B){
synchronized (C){ }
}
Thread 3:
synchronized (C){
synchronized (A){ }
}
```



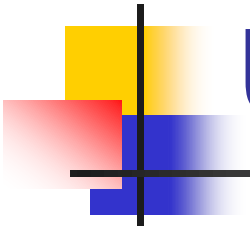
Other common type of errors

- Array index out of bounds, care needs to be taken to see that the array index values are not negative and do not exceed their bounds
- Arithmetic exceptions, include errors like divide by zero and floating point exceptions
- Off by one errors like starting variable at 1 instead of starting at 0 or vice versa, writing <= N instead of < N or vice versa etc.



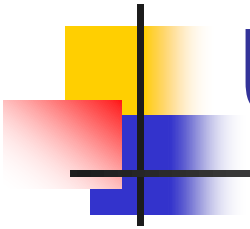
Other common type of errors..

- String handling errors like failure of string handling functions e.g strcpy, sprintf, gets etc.
- Illegal use of & instead of &&, arises if non short circuit logic (like & or |) is used instead of short circuit logic (&& or ||)



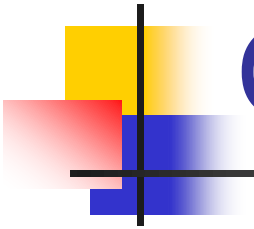
UT and Verification

- Code has to be verified before it can be used by others
- Here we discuss only verification of code written by a programmer (system verification is discussed in testing)
- There are many different techniques – two most commonly used are unit testing and inspection
- We will discuss these here



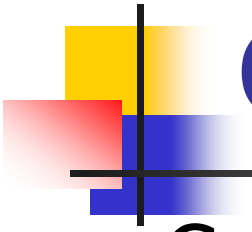
Unit Testing

- Is testing, except the focus is the module a programmer has written
- Most often UT is done by the programmer himself
- UT will require test cases for the module – will discuss in testing
- UT also requires drivers to be written to actually execute the module with test cases
- Besides the driver and test cases, tester needs to know the correct outcome as well



Code Inspections

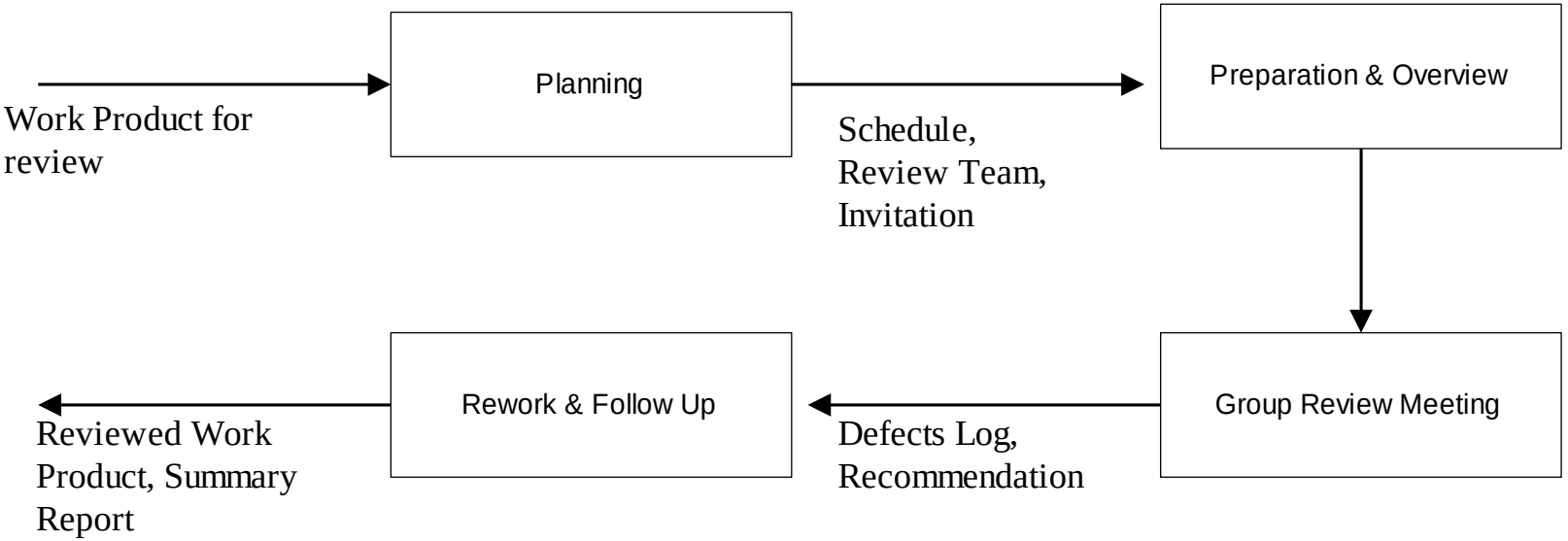
- Code inspection is another technique that is often used effectively at the unit level
- Main goal of inspection process is to detect defects in work products
- Earlier used for code, now used for all types of work products
- Is recognized as an industry best practice

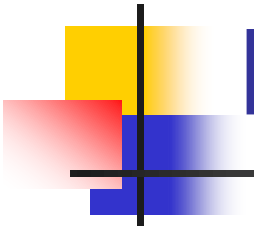


Code review…

- Conducted by group of programmers for programmers (i.e. review done by peers)
- Is a structured process with defined roles for the participants
- The focus is on identifying problems, not resolving them
- Review data is recorded and used for monitoring the effectiveness

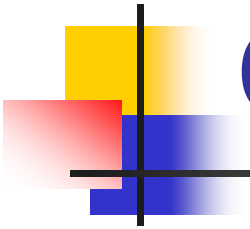
A Review Process





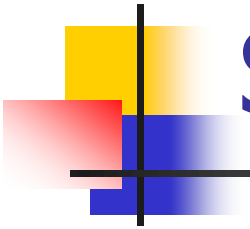
Planning

- Select the group review team – three to five people group is best
- Identify the moderator – has the main responsibility for the inspection
- Prepare package for distribution – work product for review plus supporting docs
- Package should be complete for review



Overview and Self-Review

- A brief meeting – deliver package, explain purpose of the review, intro,…
- All team members then individually review the work product
  - Lists the issues/problems they find in the self-preparation log
  - Checklists, guidelines are used
- Ideally, should be done in one sitting and issues recorded in a log



Self-Review Log

Project name:

Work product name and ID:

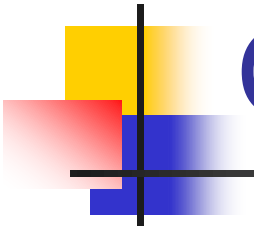
Reviewer Name

Effort spent (hours)

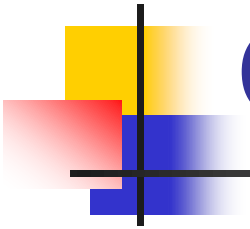
Defect list

No Location Description Criticality

Group Review Meeting

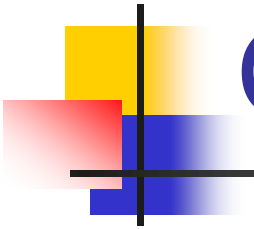


- Purpose – define the final defect list
- Entry criteria – each member has done a proper self-review (logs are reviewed)
- Group review meeting
  - A reviewer goes over the product line by line
  - At any line, all issues are raised
  - Discussion follows to identify if a defect
  - Decision recorded (by the scribe)



Group Review Meeting…

- At the end of the meeting
  - Scribe presents the list of defects/issues
  - If few defects, the work product is accepted; else it might be asked for another review
  - Group does not propose solutions – though some suggestions may be recorded
  - A summary of the inspections is prepared – useful for evaluating effectiveness



Group Review Meeting…

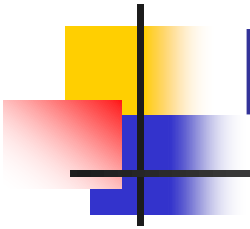
- Moderator is in-charge of the meeting and plays a central role
  - Ensures that focus is on defect detection and solutions are not discussed/proposed
  - Work product is reviewed, not the author of the work product
  - Amicable/orderly execution of the meeting
  - Uses summary report to analyze the overall effectiveness of the review

Summary Report Example

Project	XXXX
Work Product Type	Class AuctionItem
Size of work product	250 LOC of Java
Review team	P1, P2, P3
Effort (person hours)	
Preparation	3 person-hrs (total)
Group meeting	4.5 person-hrs
Total	7.5

Summary Report…

Defects	
No of major defects	3
No of minor defects	8
Total	11
Review status	Accepted
Reco for next phase	Nil
Comments	Code can be improved



Rework and Follow Up

- Defects in the defects list are fixed later by the author
- Once fixed, author gets it OKed by the moderator, or goes for another review
- Once all defects/issues are satisfactorily addressed, review is completed and collected data is submitted