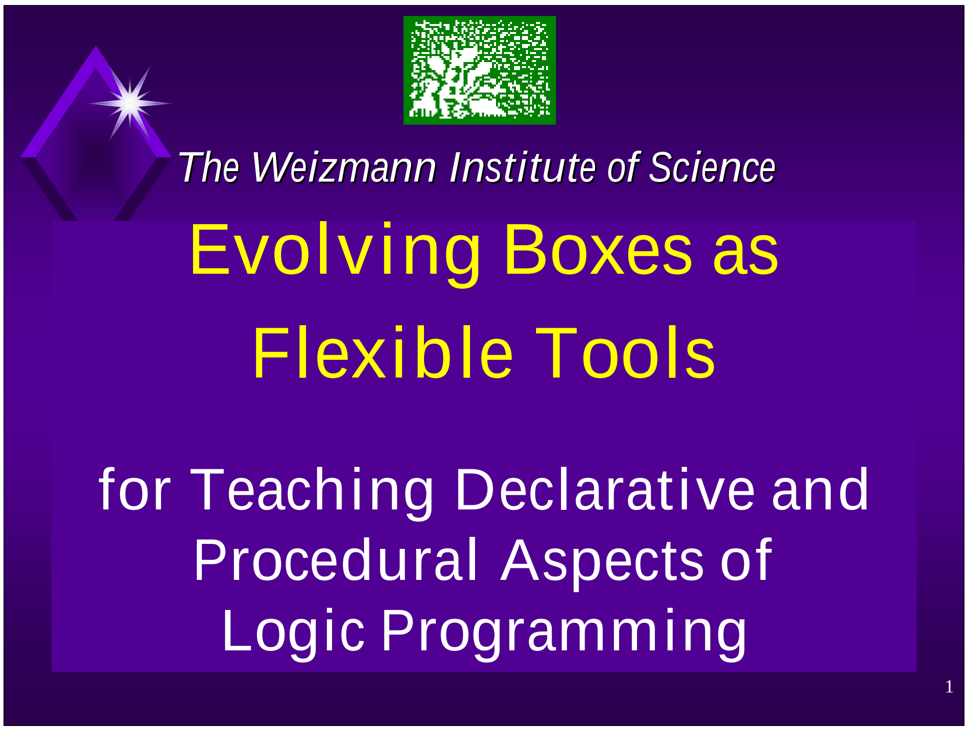
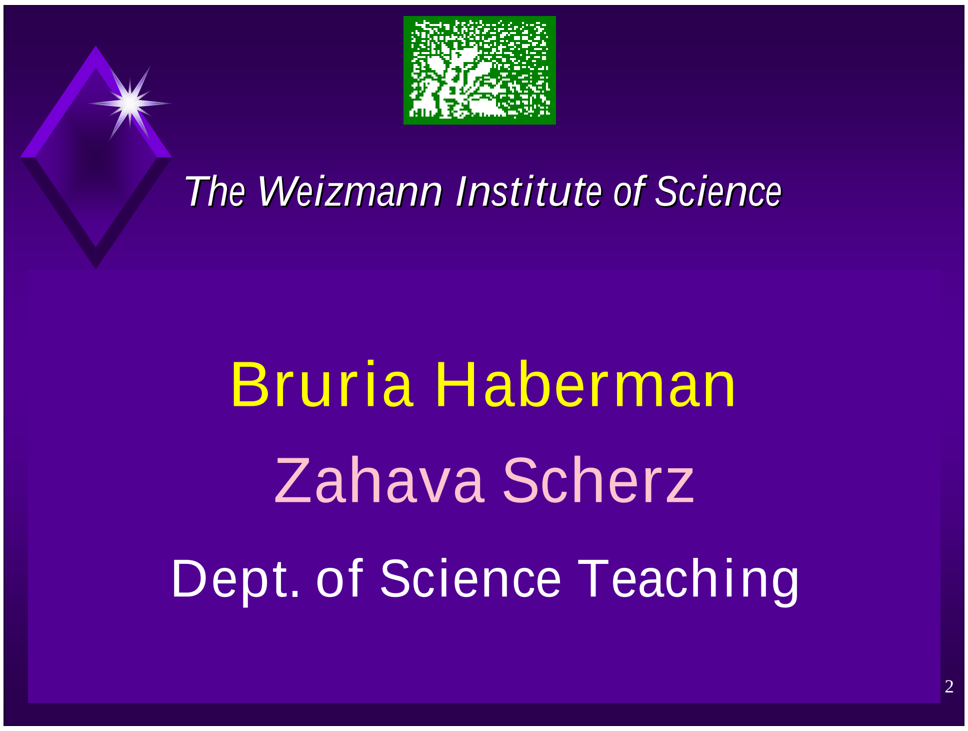


The Weizmann Institute of Science

Evolving Boxes as Flexible Tools

for Teaching Declarative and Procedural Aspects of Logic Programming

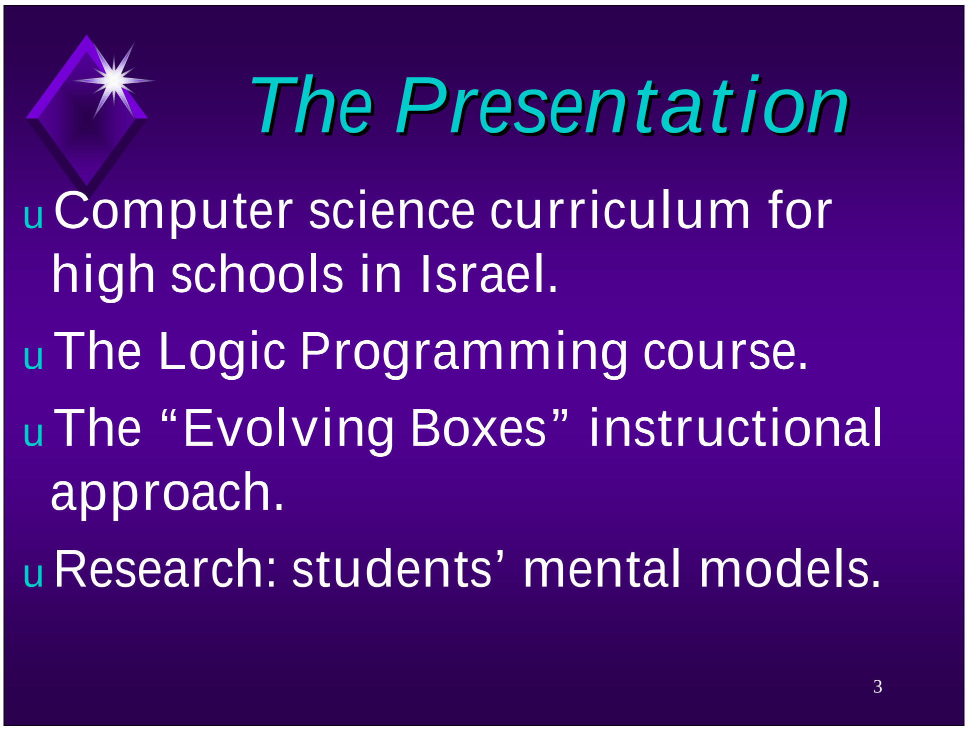


The Weizmann Institute of Science

Bruria Haberman

Zahava Scherz

Dept. of Science Teaching

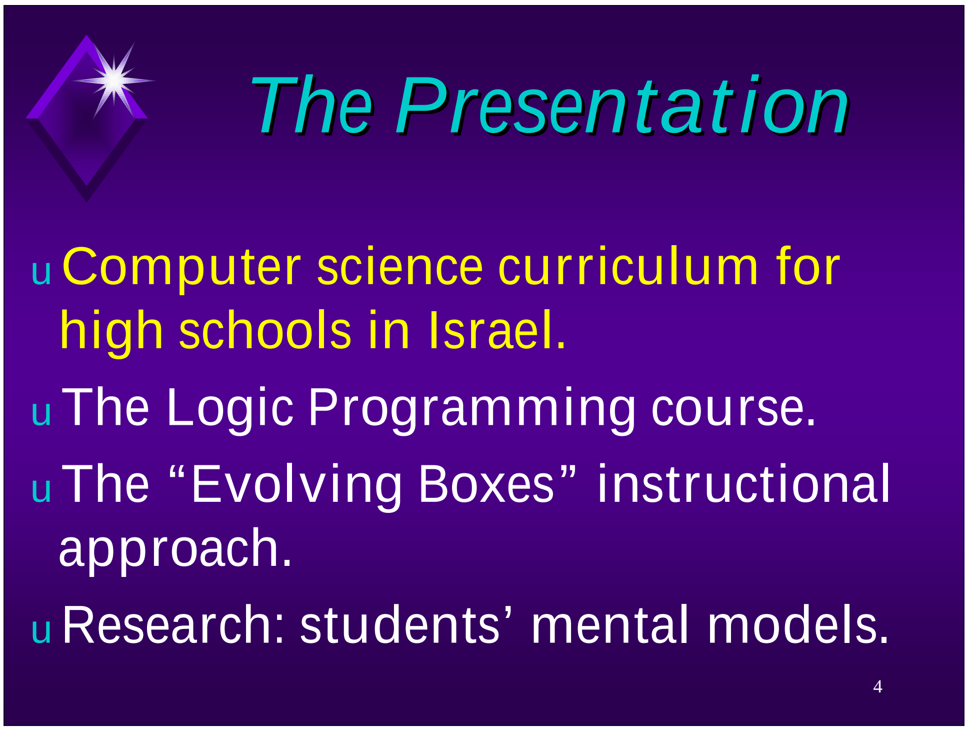


The Presentation

- Computer science curriculum for high schools in Israel.
- The Logic Programming course.
- The “Evolving Boxes” instructional approach.
- Research: students’ mental models.

3

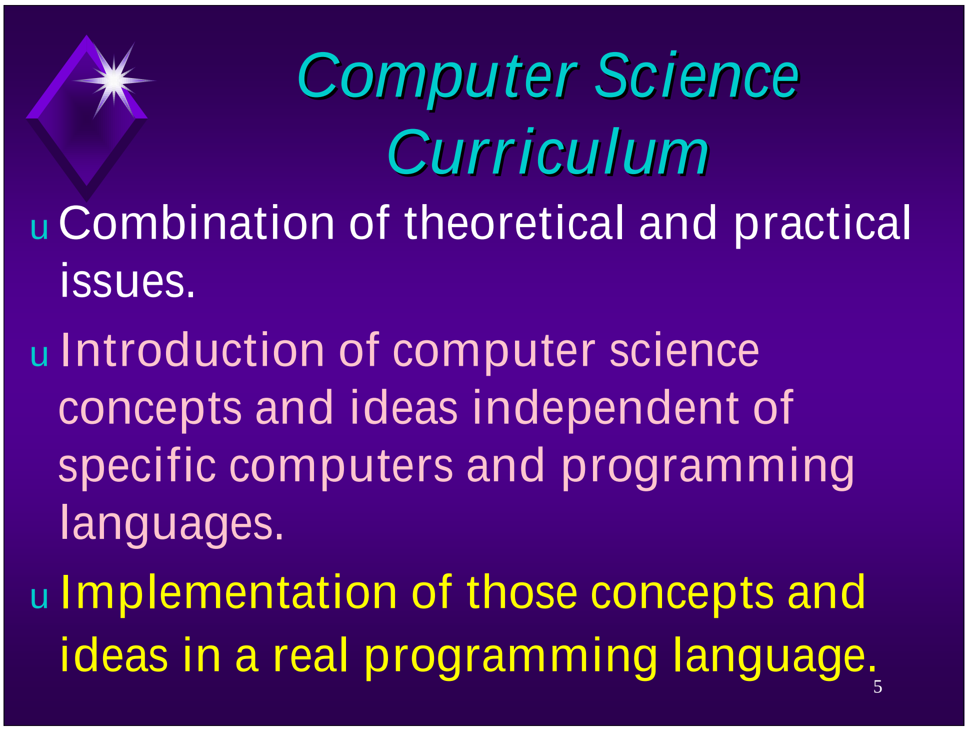
2



The Presentation

- Computer science curriculum for high schools in Israel.
- The Logic Programming course.
- The “Evolving Boxes” instructional approach.
- Research: students’ mental models.

4

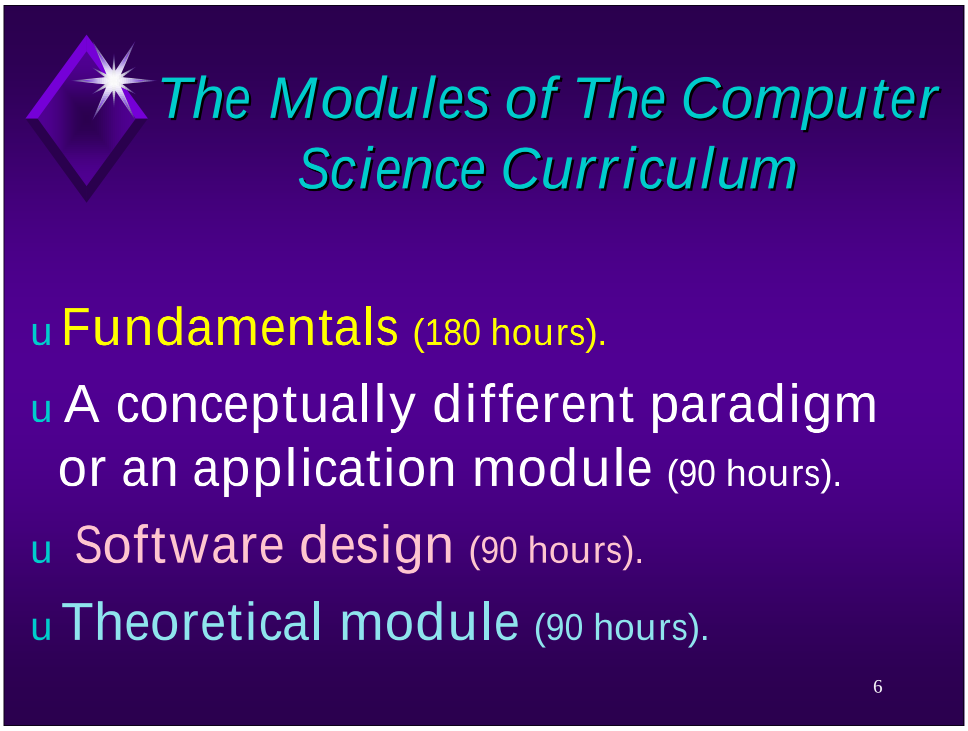


Computer Science Curriculum

- Combination of theoretical and practical issues.
- Introduction of computer science concepts and ideas independent of specific computers and programming languages.
- Implementation of those concepts and ideas in a real programming language.

5

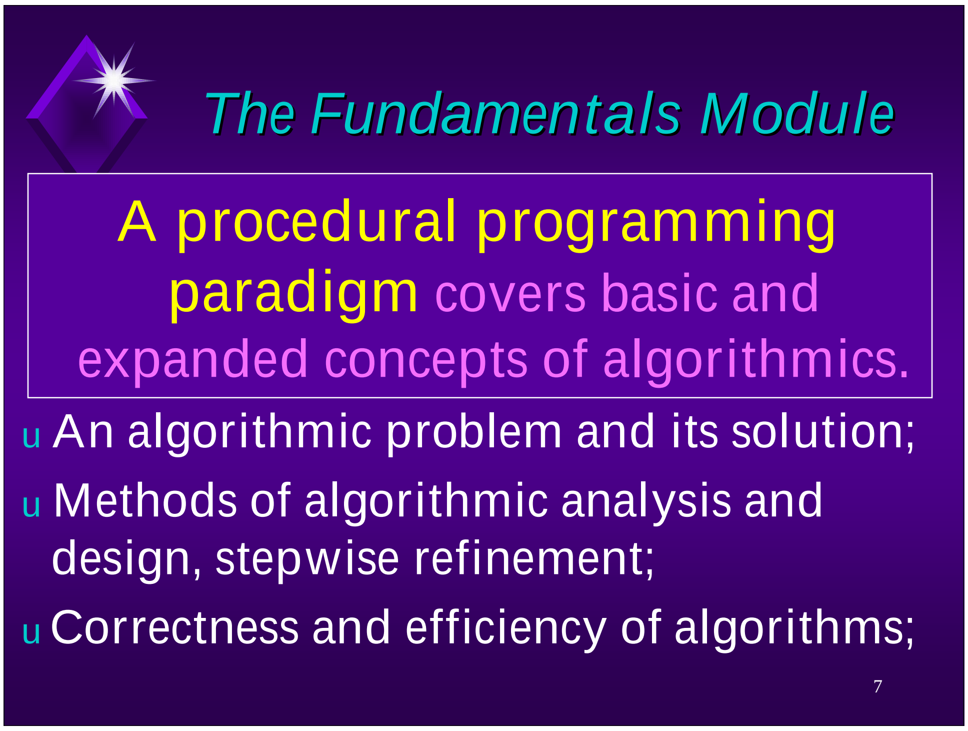
3



The Modules of The Computer Science Curriculum

- Fundamentals (180 hours).
- A conceptually different paradigm or an application module (90 hours).
- Software design (90 hours).
- Theoretical module (90 hours).

6



The Fundamentals Module

A procedural programming paradigm covers basic and expanded concepts of algorithmics.

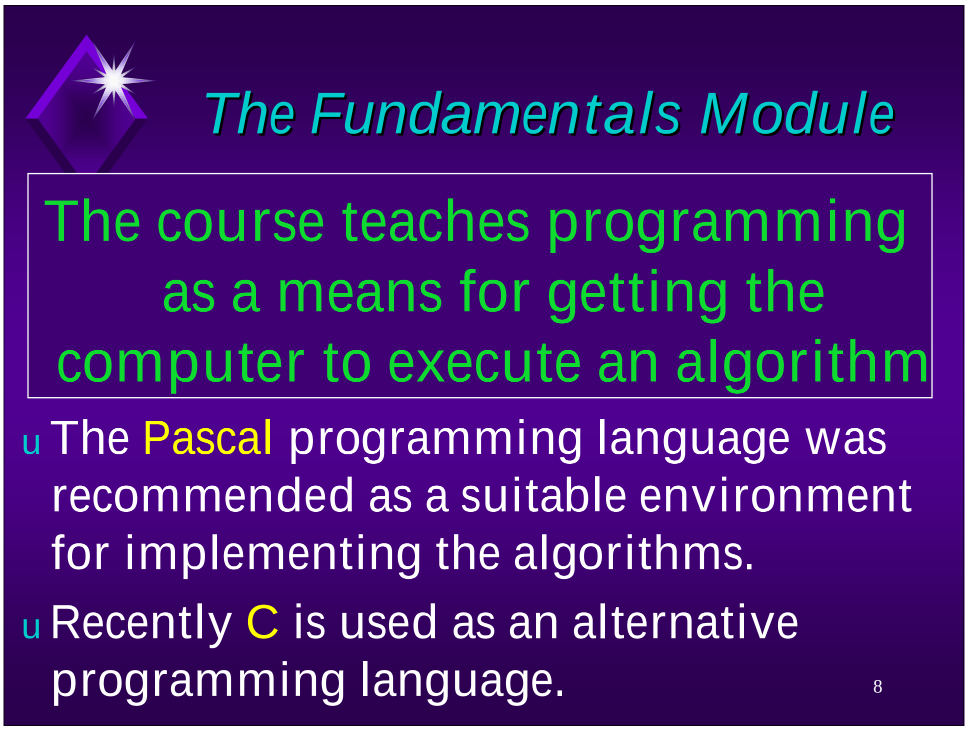
u An algorithmic problem and its solution;

u Methods of algorithmic analysis and design, stepwise refinement;

u Correctness and efficiency of algorithms;

7

4



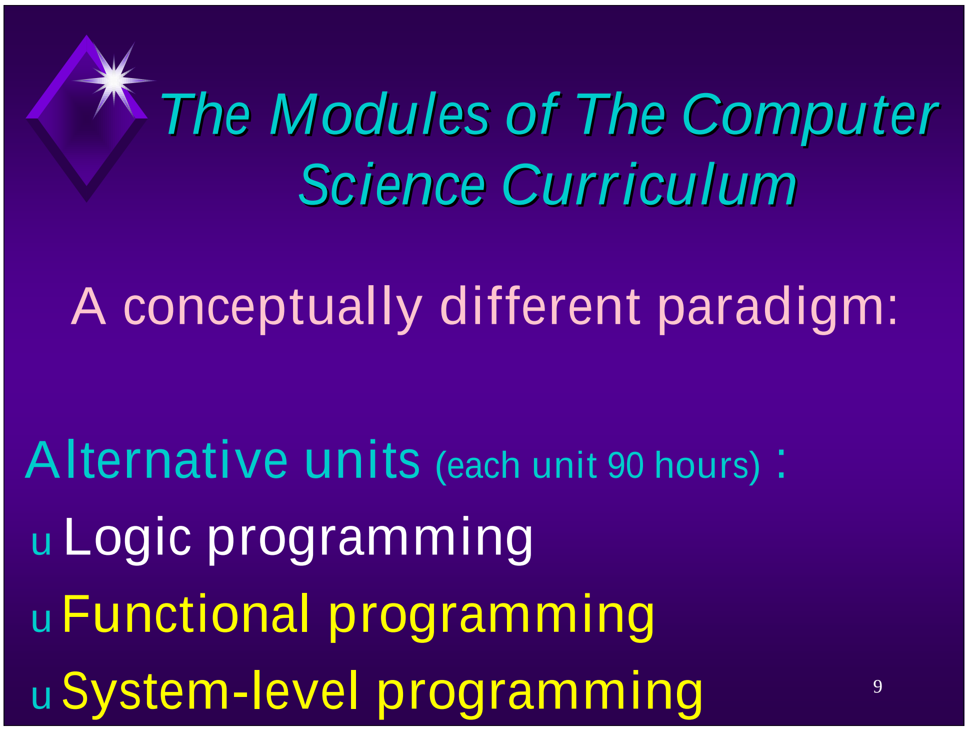
The Fundamentals Module

The course teaches programming as a means for getting the computer to execute an algorithm

u The Pascal programming language was recommended as a suitable environment for implementing the algorithms.

u Recently C is used as an alternative programming language.

8



The Modules of The Computer Science Curriculum

A conceptually different paradigm:

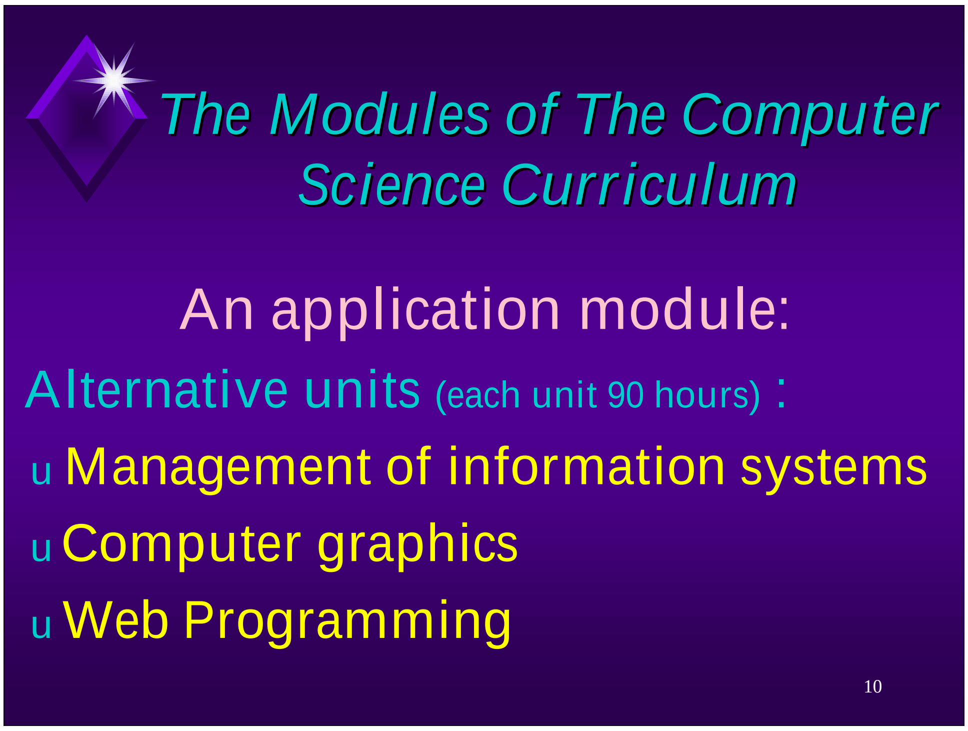
Alternative units (each unit 90 hours) :

u Logic programming

u Functional programming

u System-level programming

9



The Modules of The Computer Science Curriculum

An application module:

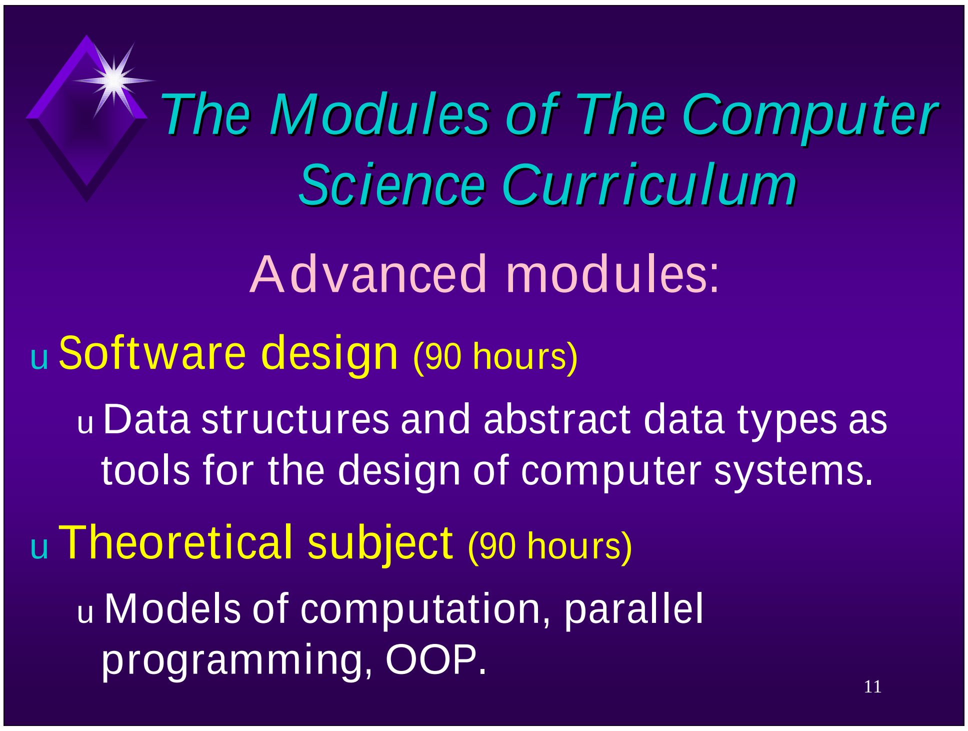
Alternative units (each unit 90 hours) :

u Management of information systems

u Computer graphics

u Web Programming

10



The Modules of The Computer Science Curriculum

Advanced modules:

u Software design (90 hours)

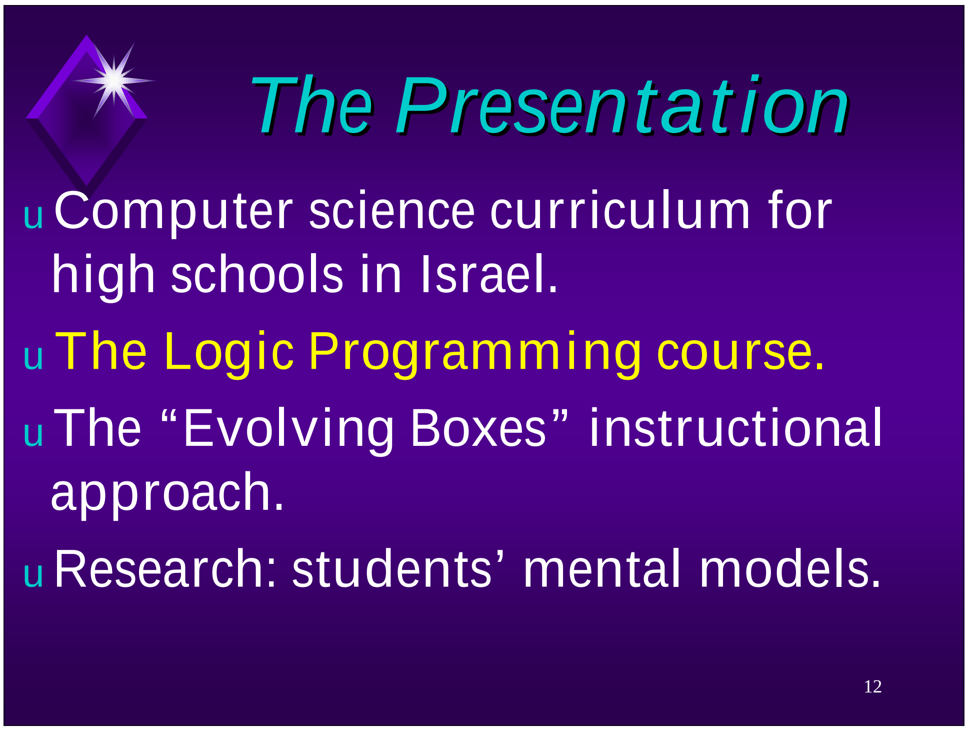
u Data structures and abstract data types as tools for the design of computer systems.

u Theoretical subject (90 hours)

u Models of computation, parallel programming, OOP.

11

6



The Presentation

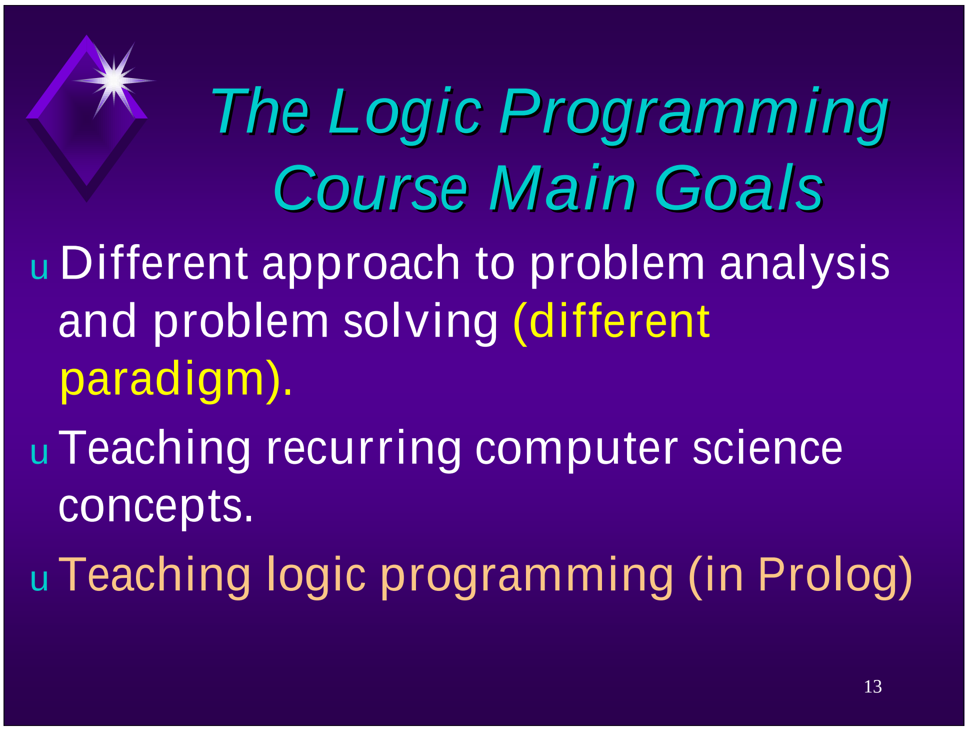
u Computer science curriculum for high schools in Israel.

u The Logic Programming course.

u The “Evolving Boxes” instructional approach.

u Research: students’ mental models.

12



The Logic Programming Course Main Goals

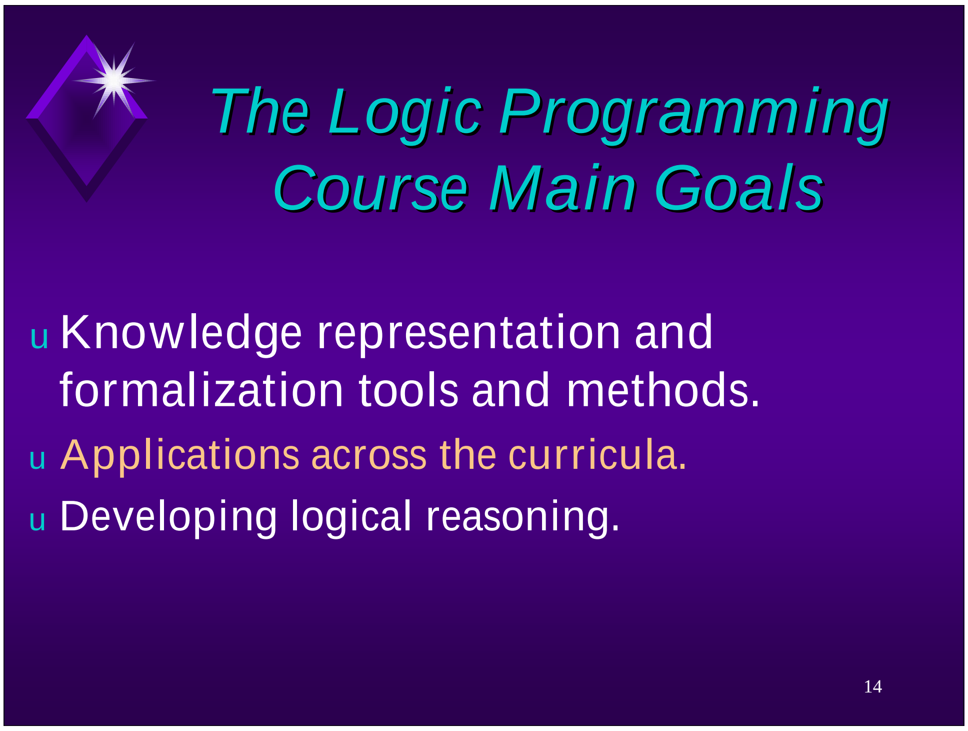
u Different approach to problem analysis and problem solving (different paradigm).

u Teaching recurring computer science concepts.

u Teaching logic programming (in Prolog)

13

7



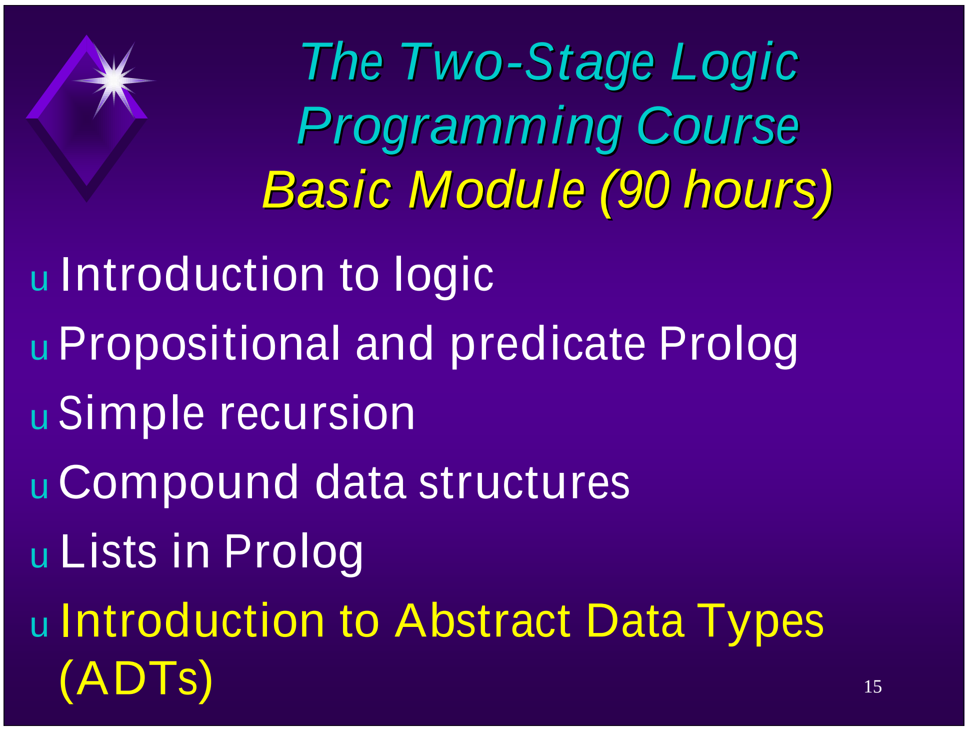
The Logic Programming Course Main Goals

u Knowledge representation and formalization tools and methods.

u Applications across the curricula.

u Developing logical reasoning.

14



The Two-Stage Logic Programming Course
Basic Module (90 hours)

u Introduction to logic

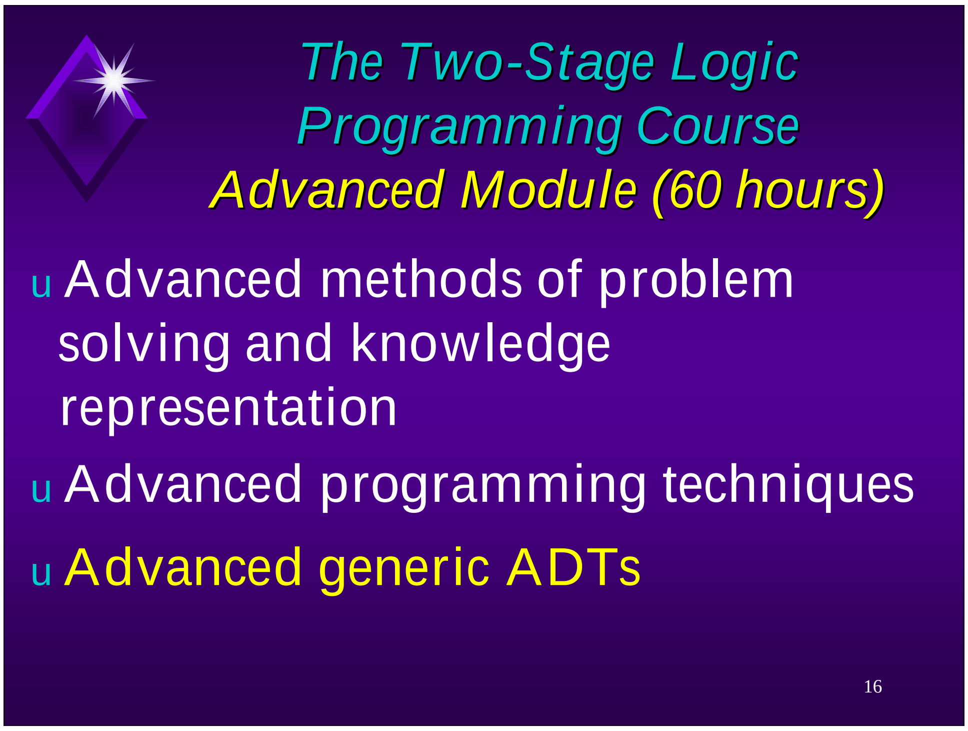
u Propositional and predicate Prolog

u Simple recursion

u Compound data structures

u Lists in Prolog

u Introduction to Abstract Data Types (ADTs)

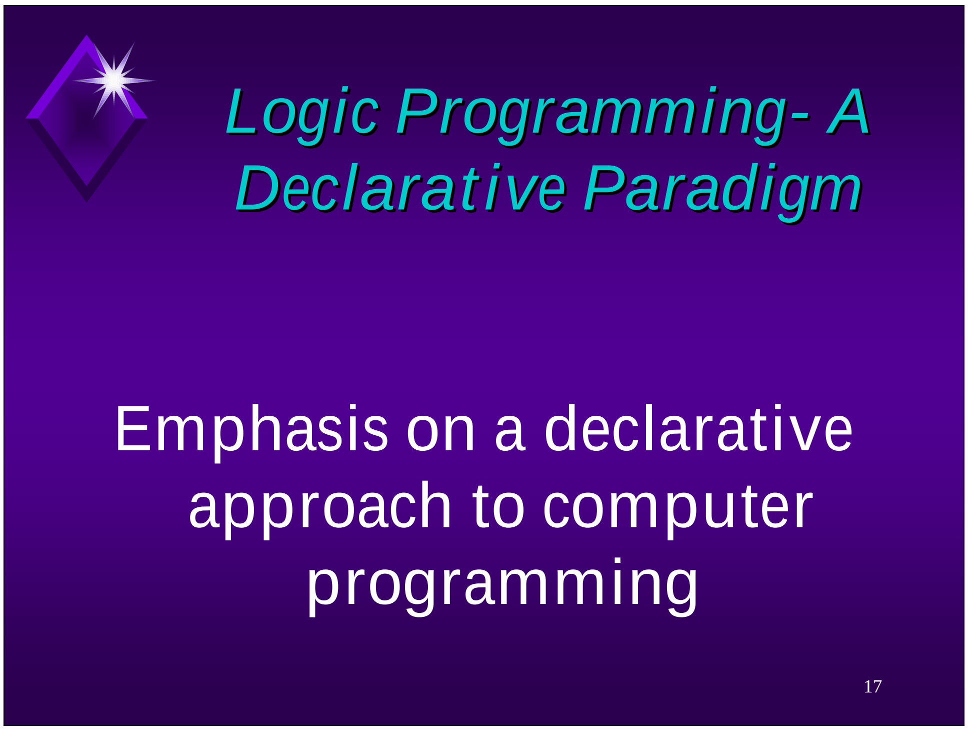


The Two-Stage Logic Programming Course
Advanced Module (60 hours)

u Advanced methods of problem solving and knowledge representation

u Advanced programming techniques

u Advanced generic ADTs

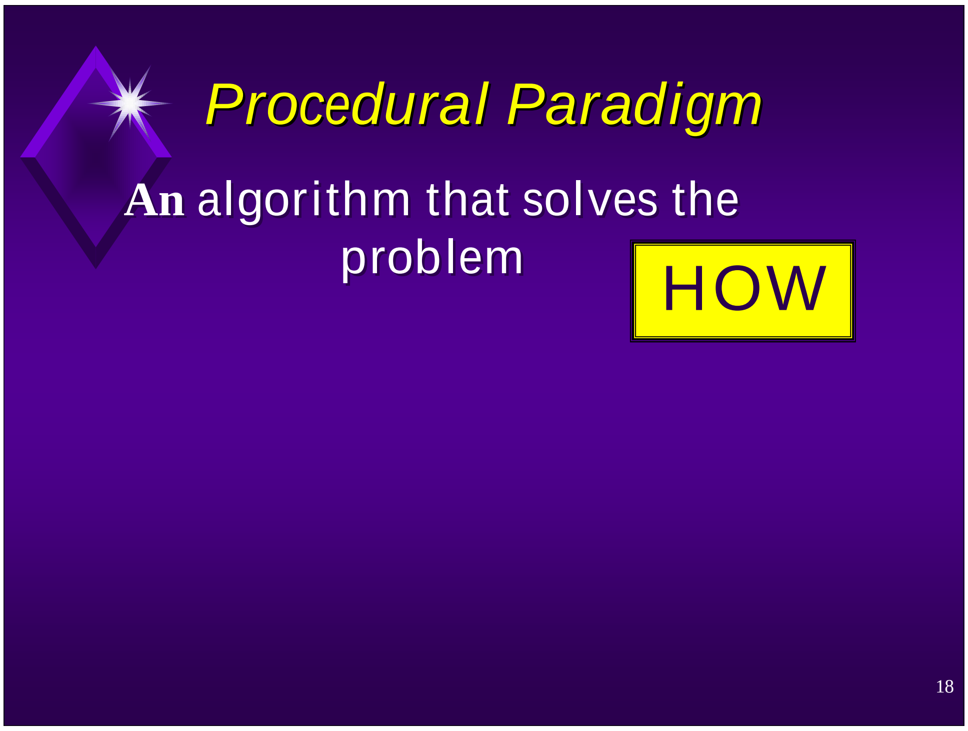


Logic Programming- A Declarative Paradigm

Emphasis on a declarative approach to computer programming

17

9

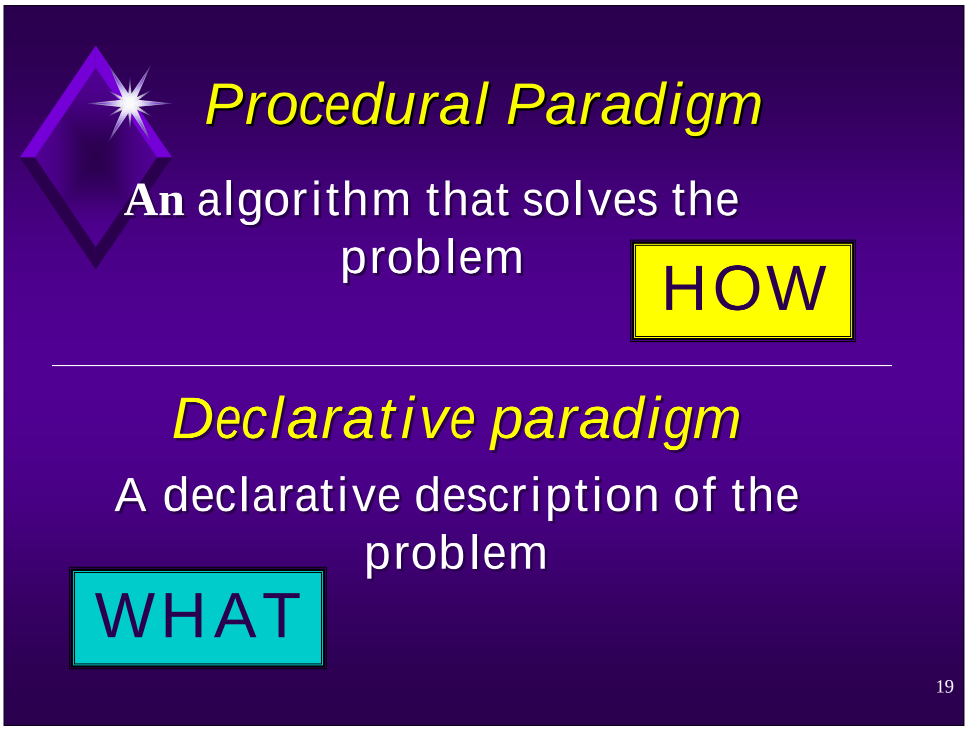


Procedural Paradigm

An algorithm that solves the problem

HOW

18



Procedural Paradigm

An algorithm that solves the problem

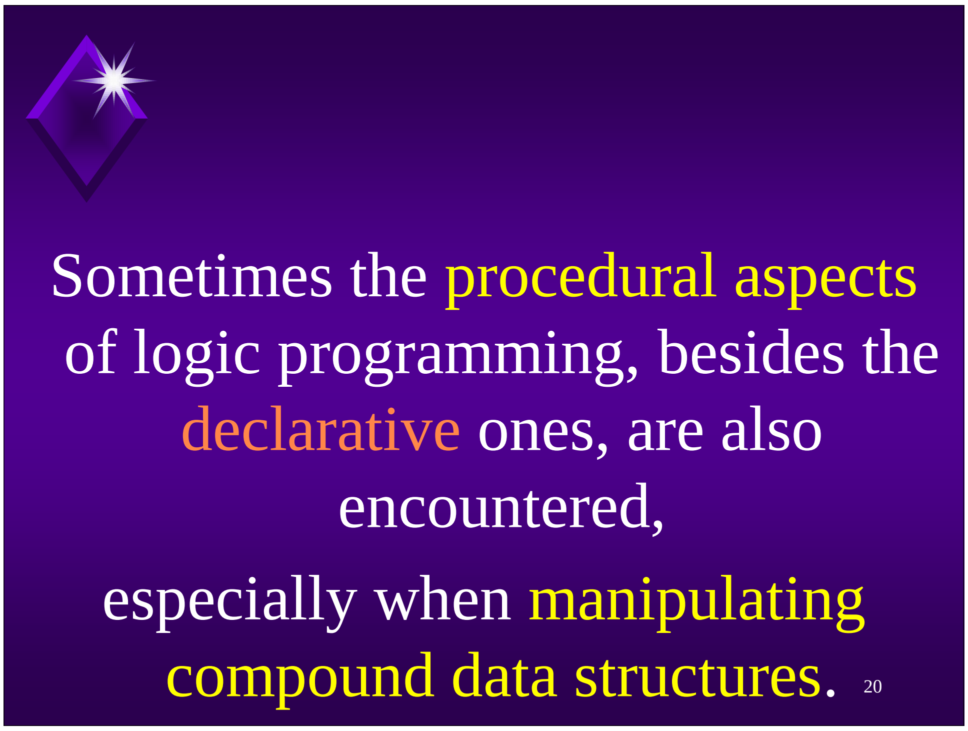
HOW

Declarative paradigm

A declarative description of the problem

WHAT

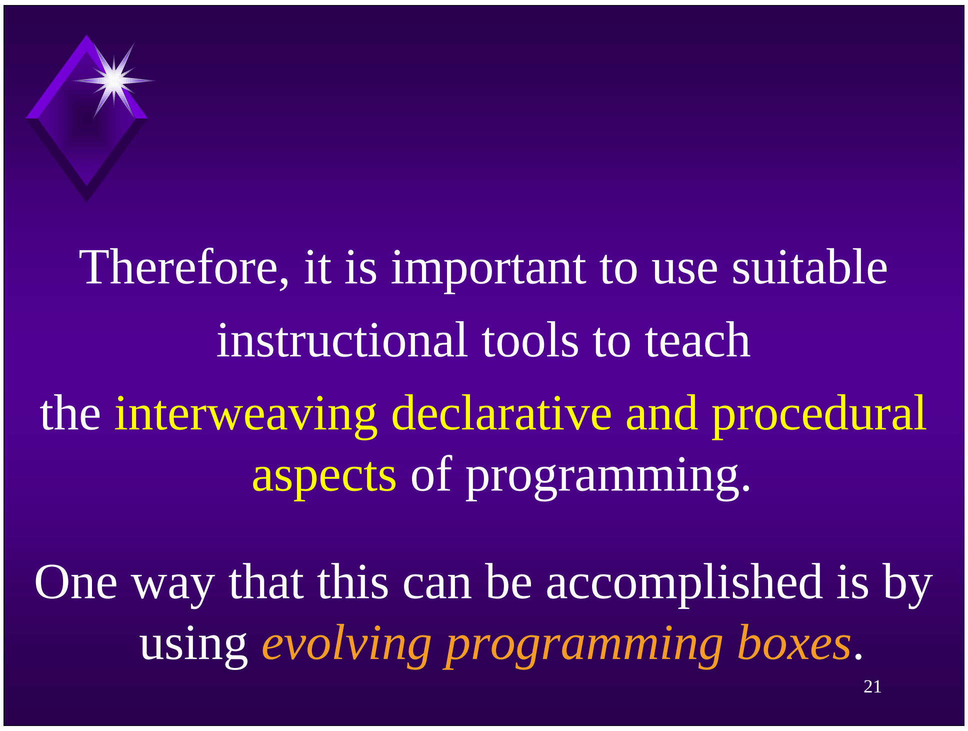
19



Sometimes the procedural aspects of logic programming, besides the declarative ones, are also encountered,

especially when manipulating compound data structures.

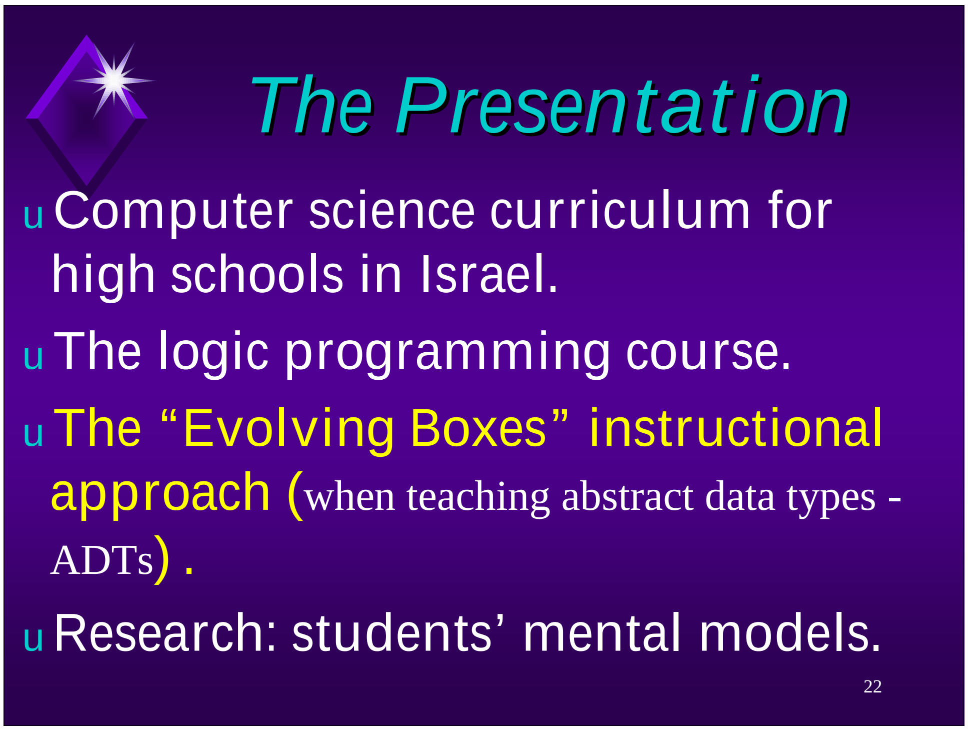
20


Therefore, it is important to use suitable instructional tools to teach the interweaving declarative and procedural aspects of programming.

One way that this can be accomplished is by using *evolving programming boxes*.

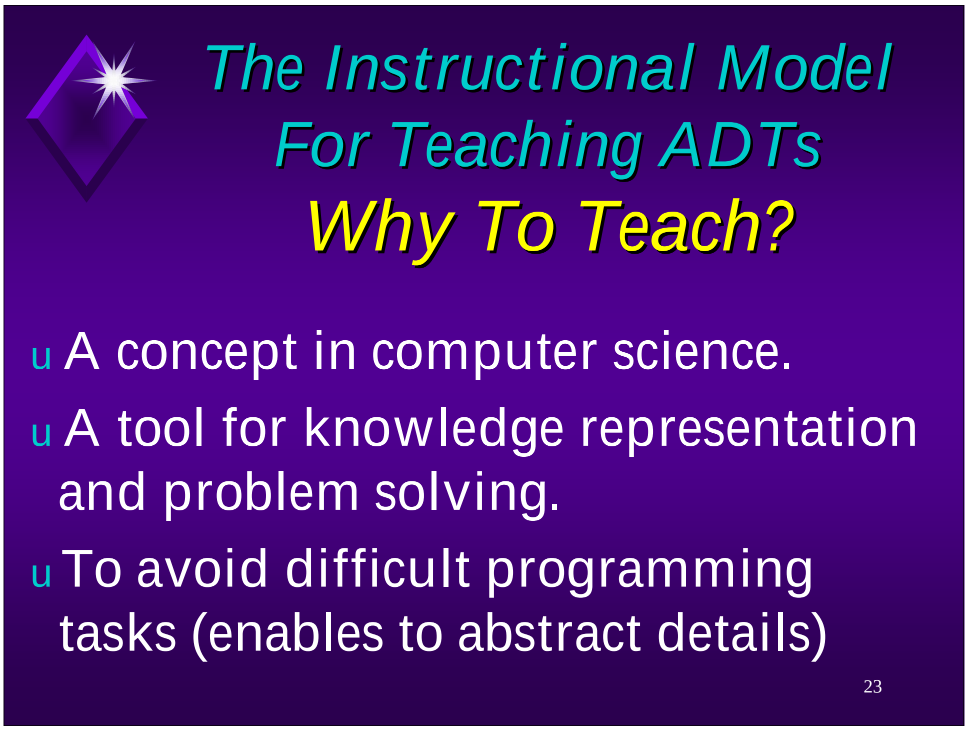
21

11


The Presentation

- Computer science curriculum for high schools in Israel.
- The logic programming course.
- The “Evolving Boxes” instructional approach (when teaching abstract data types - ADTs) .
- Research: students’ mental models.

22



The Instructional Model For Teaching ADTs

Why To Teach?

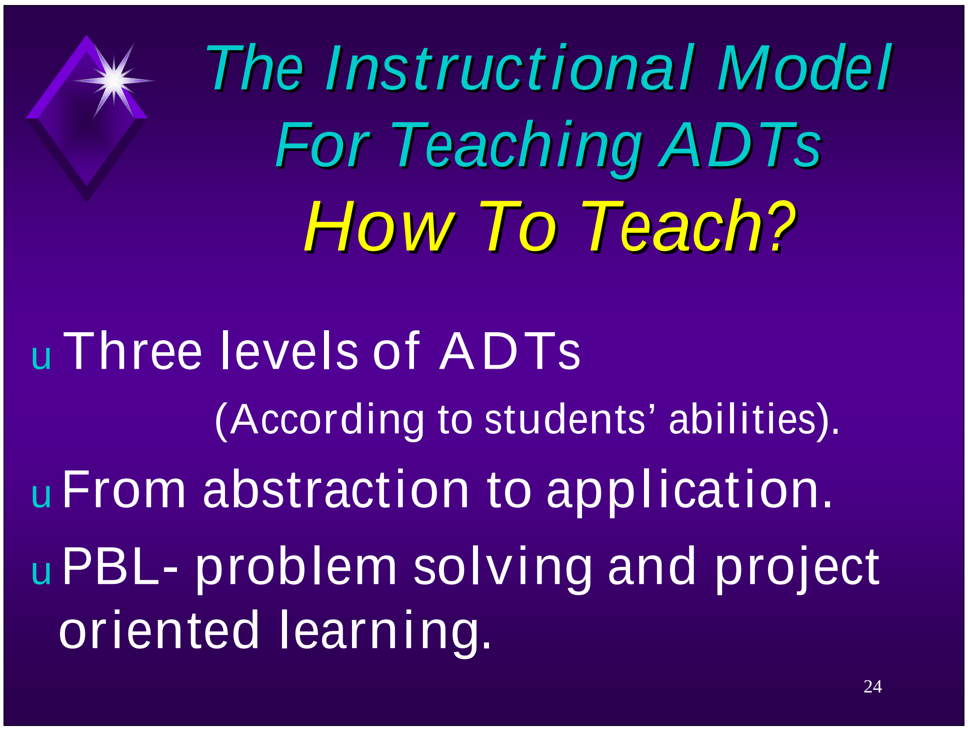
u A concept in computer science.

u A tool for knowledge representation and problem solving.

u To avoid difficult programming tasks (enables to abstract details)

23

12



The Instructional Model For Teaching ADTs

How To Teach?

u Three levels of ADTs

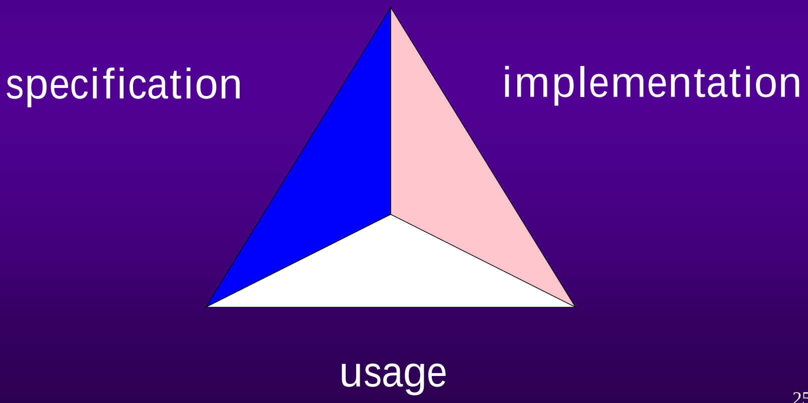
(According to students' abilities).

u From abstraction to application.

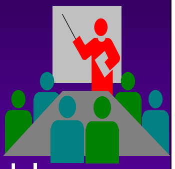
u PBL- problem solving and project oriented learning.

24

The Levels of Abstract Data Type



25



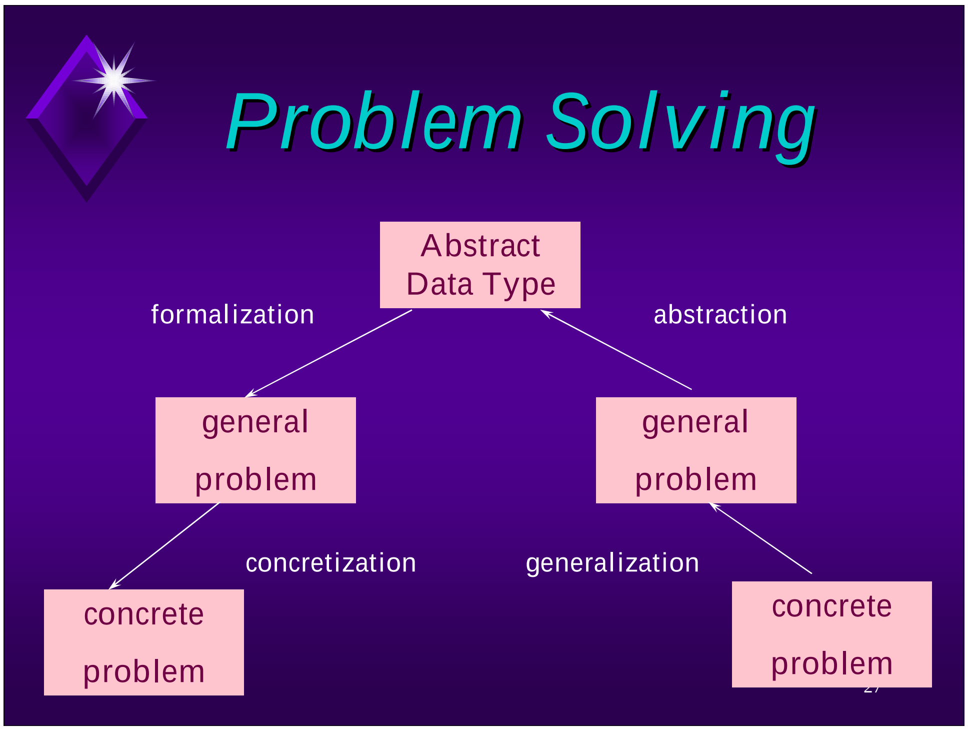
- **Abstraction**

  An introduction to abstract and formal concepts and to various methods of problem solving and knowledge representation.
- **Implementation**

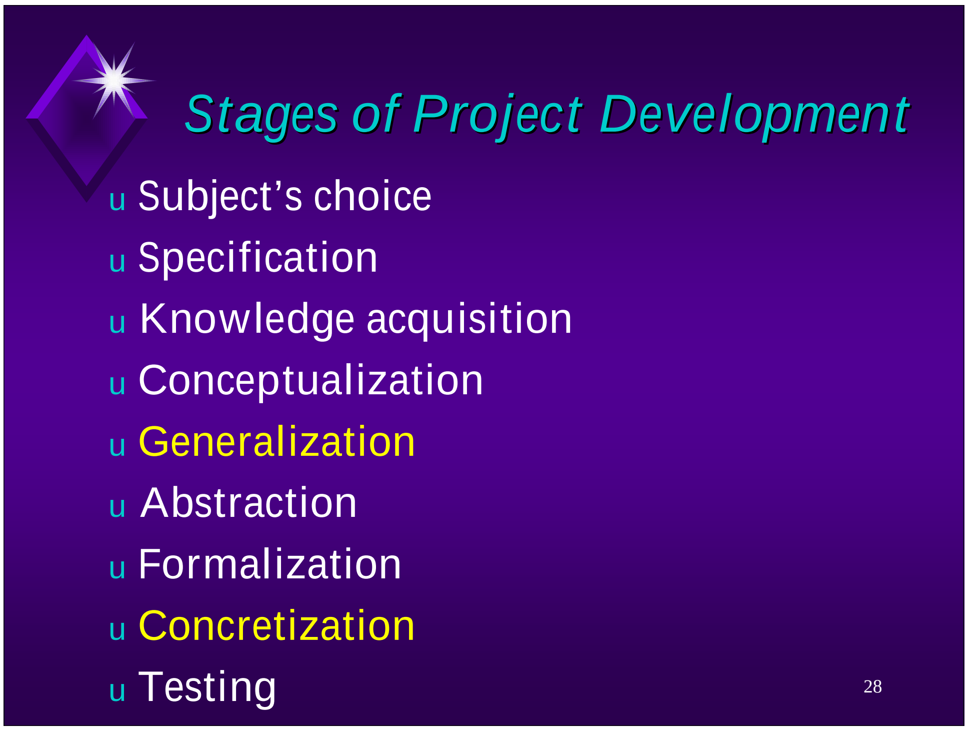
  Implementation of abstract and formal concepts in terms of computer programs.
- **Application**

  Problem solving and development of computer knowledge-based systems.

26



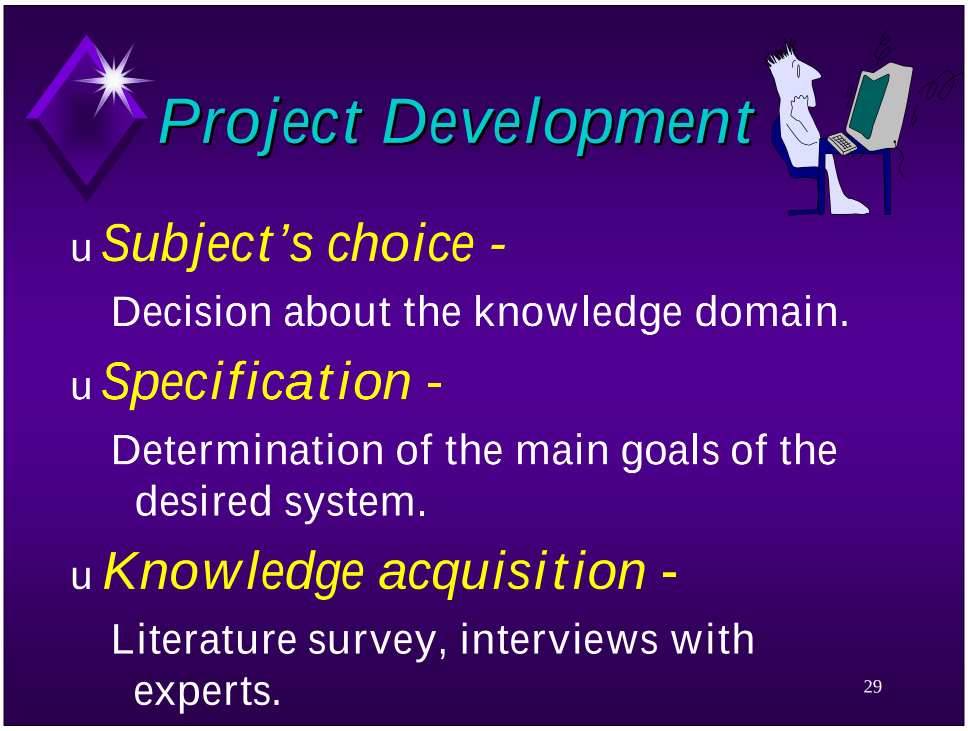
Problem Solving



Stages of Project Development

- Subject’s choice
- Specification
- Knowledge acquisition
- Conceptualization
- Generalization
- Abstraction
- Formalization
- Concretization
- Testing

28



Project Development

- *Subject's choice -*

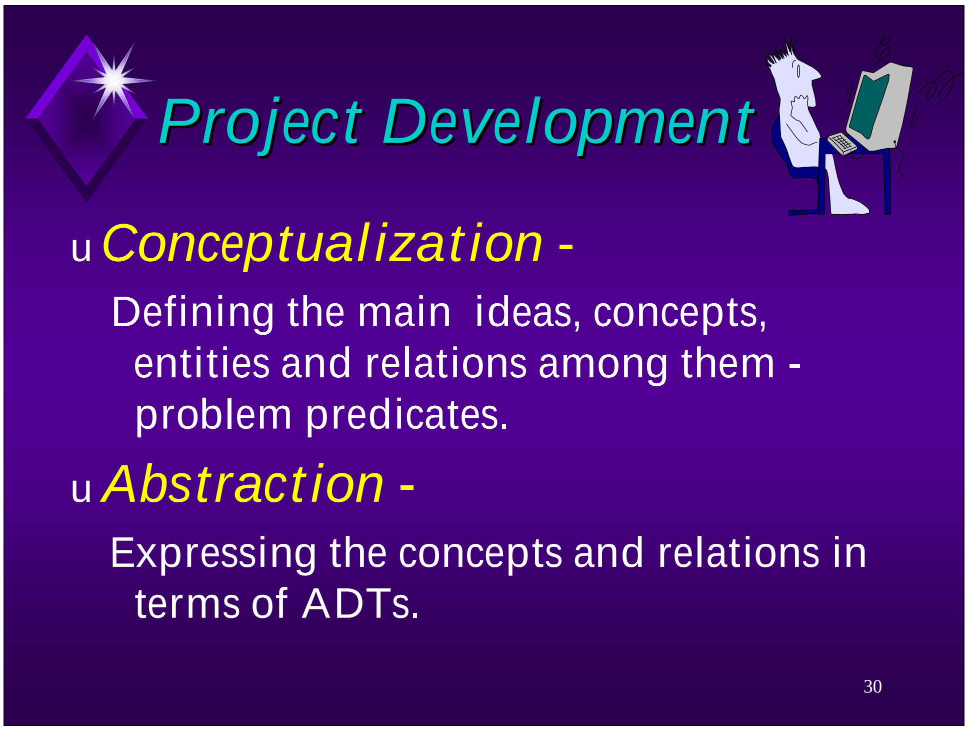
  Decision about the knowledge domain.
- *Specification -*

  Determination of the main goals of the desired system.
- *Knowledge acquisition -*

  Literature survey, interviews with experts.

29

15



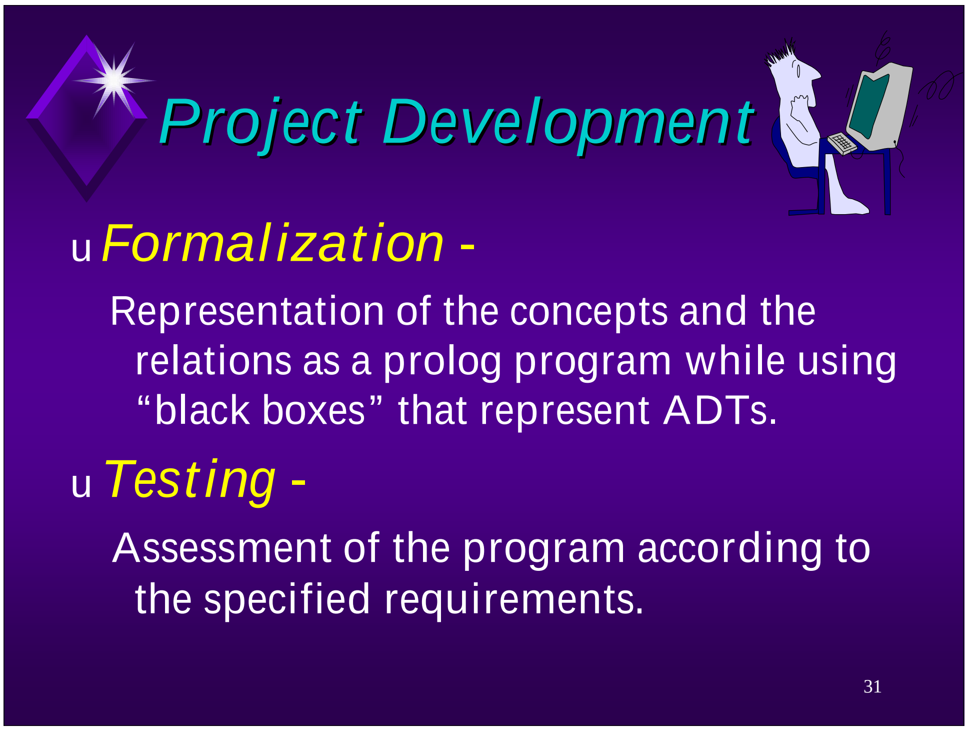
Project Development

- *Conceptualization -*

  Defining the main ideas, concepts, entities and relations among them - problem predicates.
- *Abstraction -*

  Expressing the concepts and relations in terms of ADTs.

30



Project Development

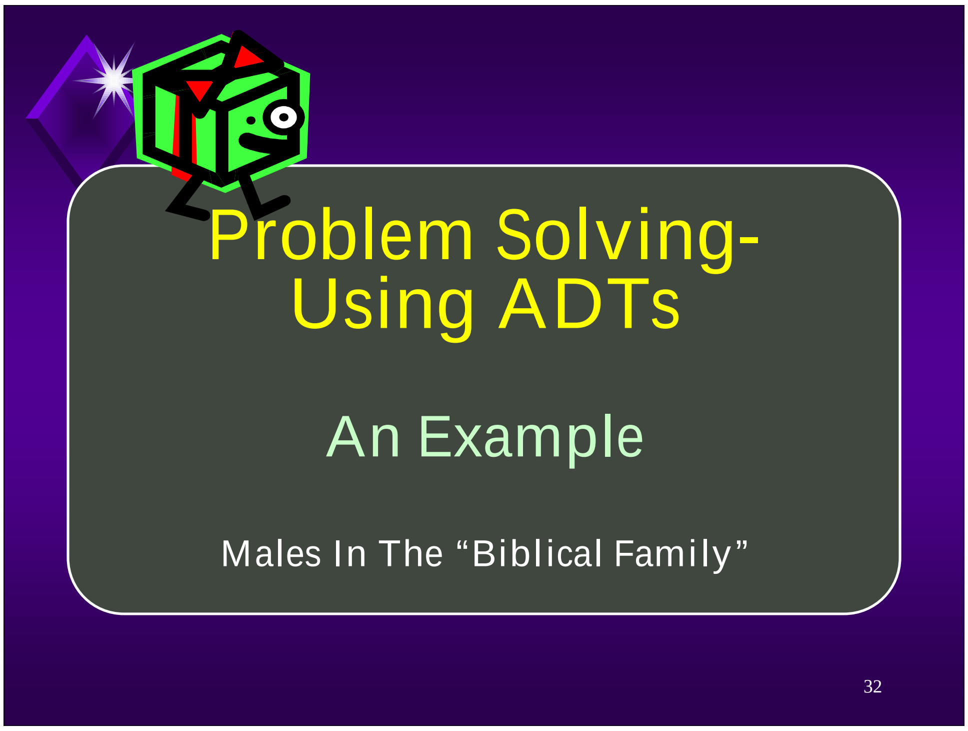
- *Formalization -*

  Representation of the concepts and the relations as a prolog program while using "black boxes" that represent ADTs.

- *Testing -*

  Assessment of the program according to the specified requirements.

31



Problem Solving- Using ADTs

An Example

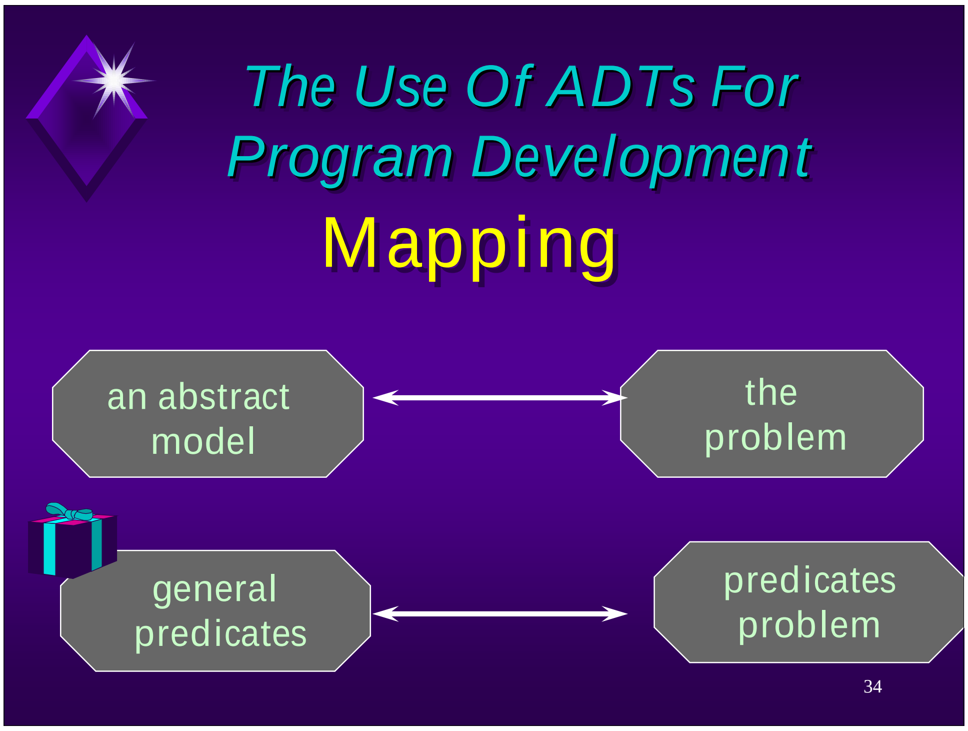
Males In The "Biblical Family"

32

Abstraction

33

17

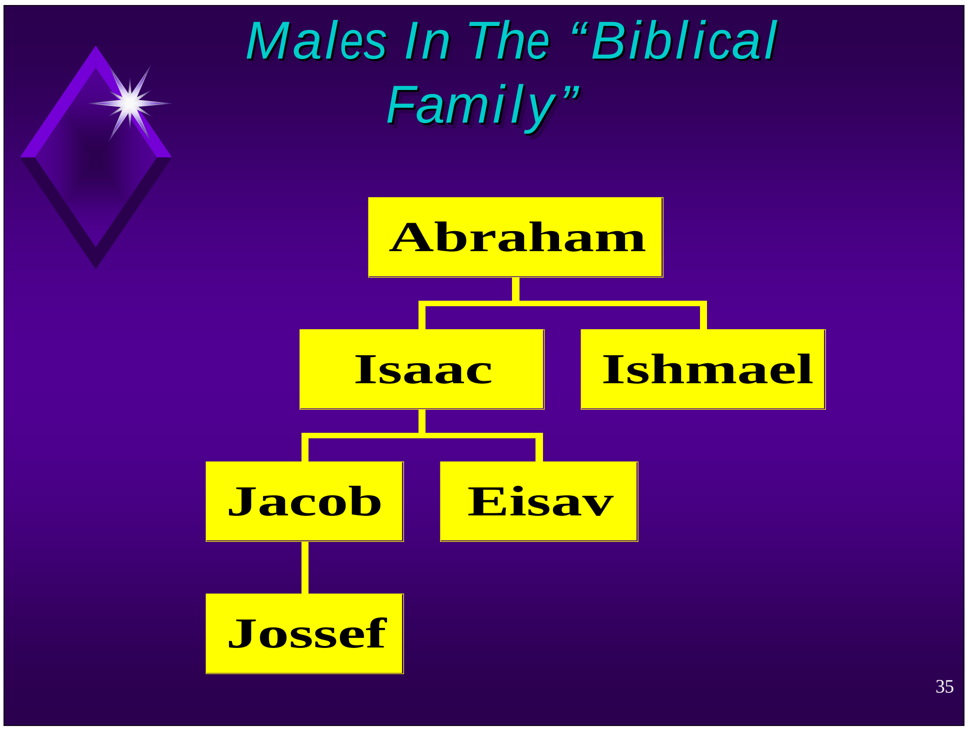


The Use Of ADTs For Program Development

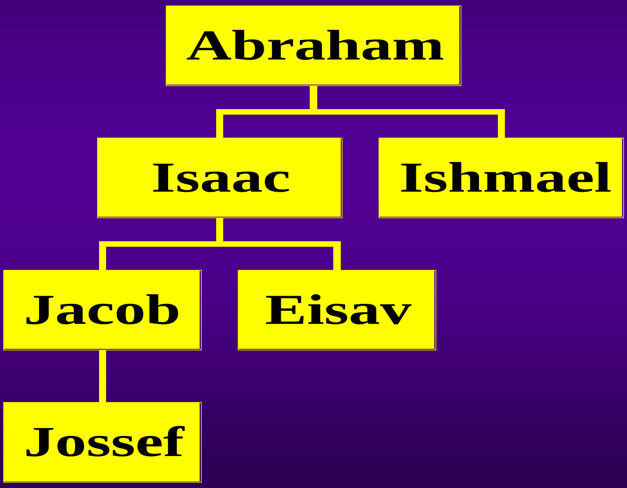
Mapping

$$\begin{tikzcd}
\text{an abstract model} \arrow[r, leftrightarrow] & \text{the problem} \\
\text{general predicates} \arrow[r, leftrightarrow] & \text{predicates problem}
\end{tikzcd}$$

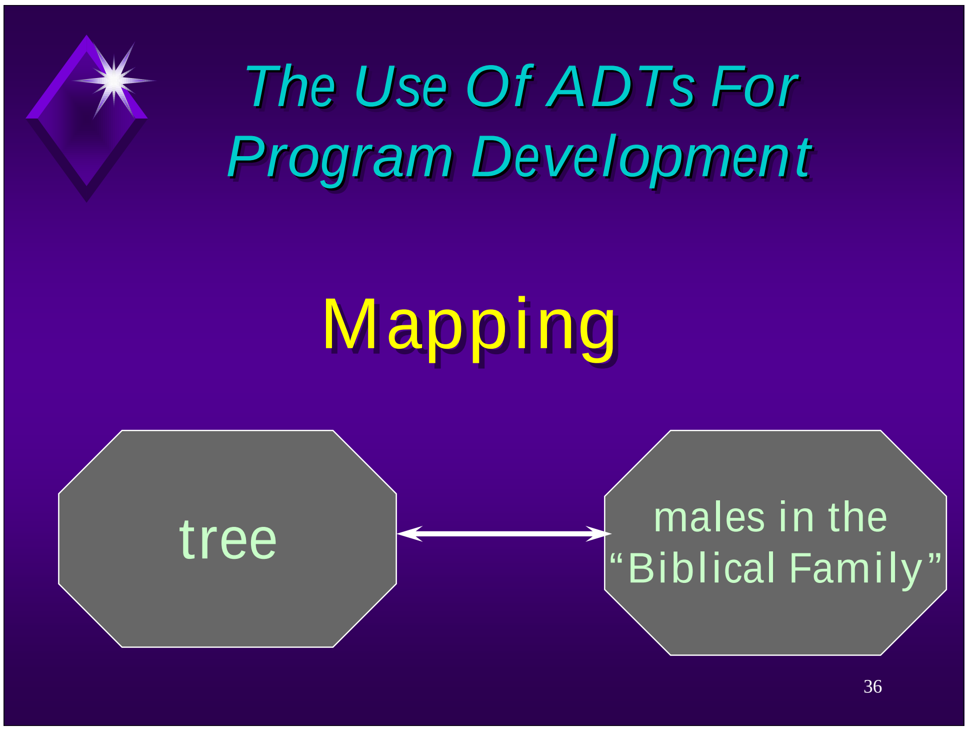
34



Males In The "Biblical Family"

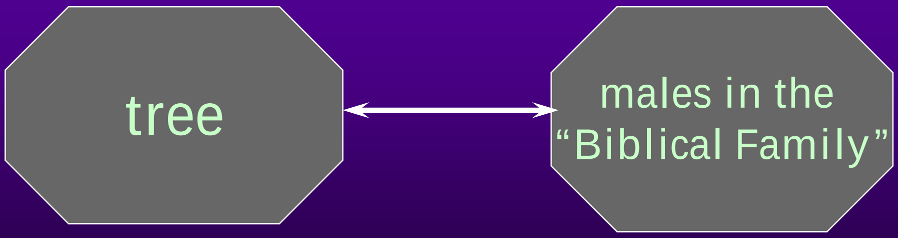


35

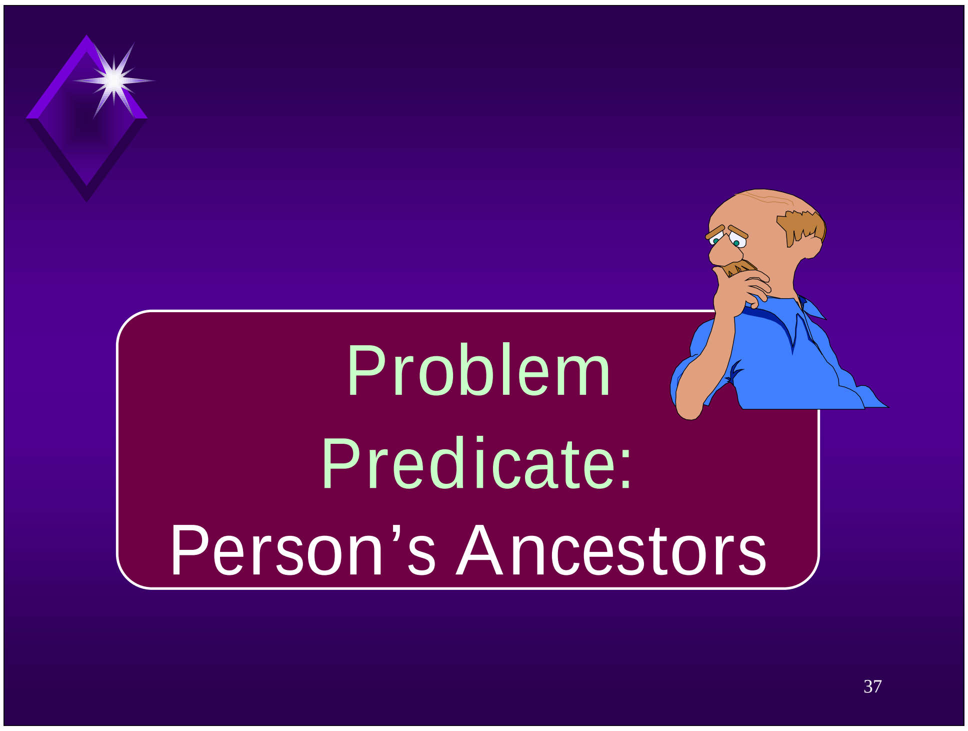


The Use Of ADTs For Program Development

Mapping



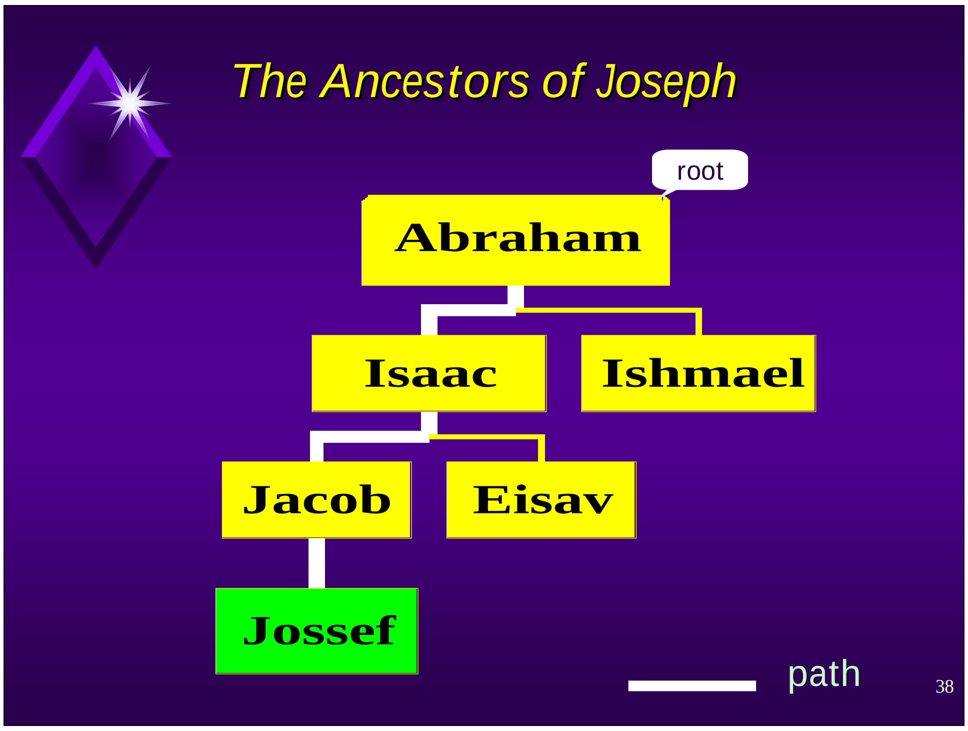
36



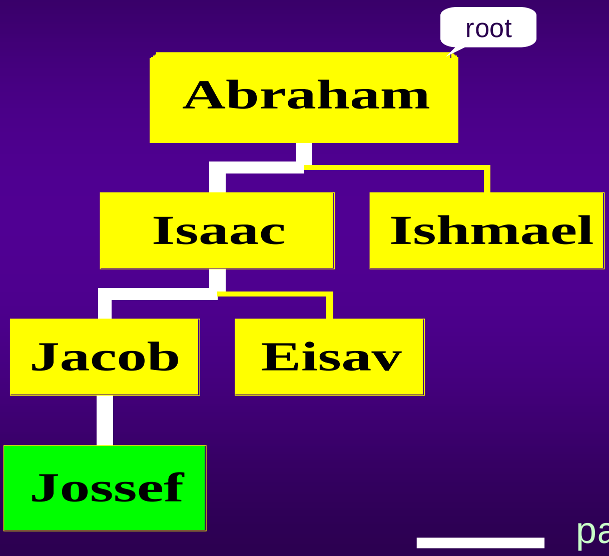
Problem Predicate: Person’s Ancestors

37

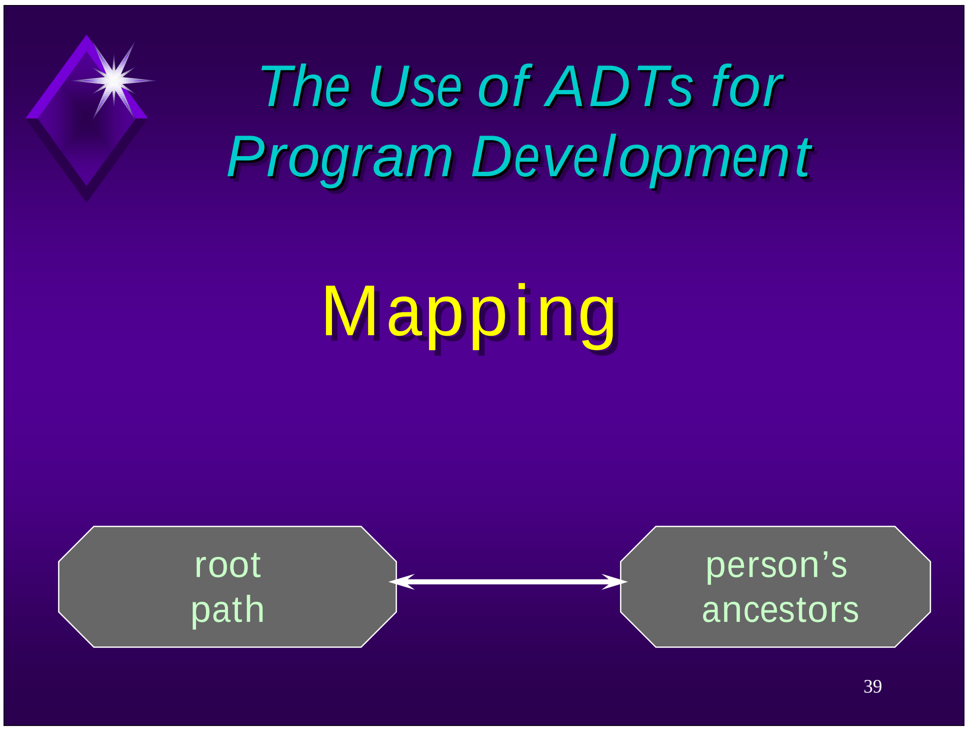
19



The Ancestors of Joseph



38



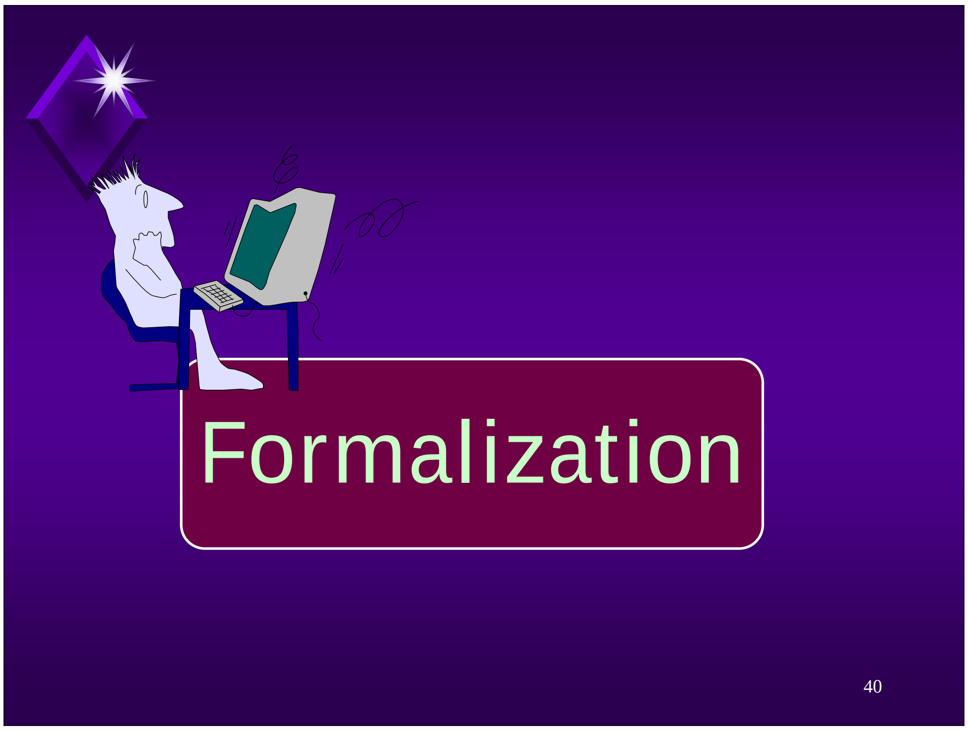
The Use of ADTs for Program Development

Mapping

$$\begin{tikzcd}\boxed{\text{root path}} \arrow[r, leftrightarrow] & \boxed{\text{person's ancestors}}\end{tikzcd}$$

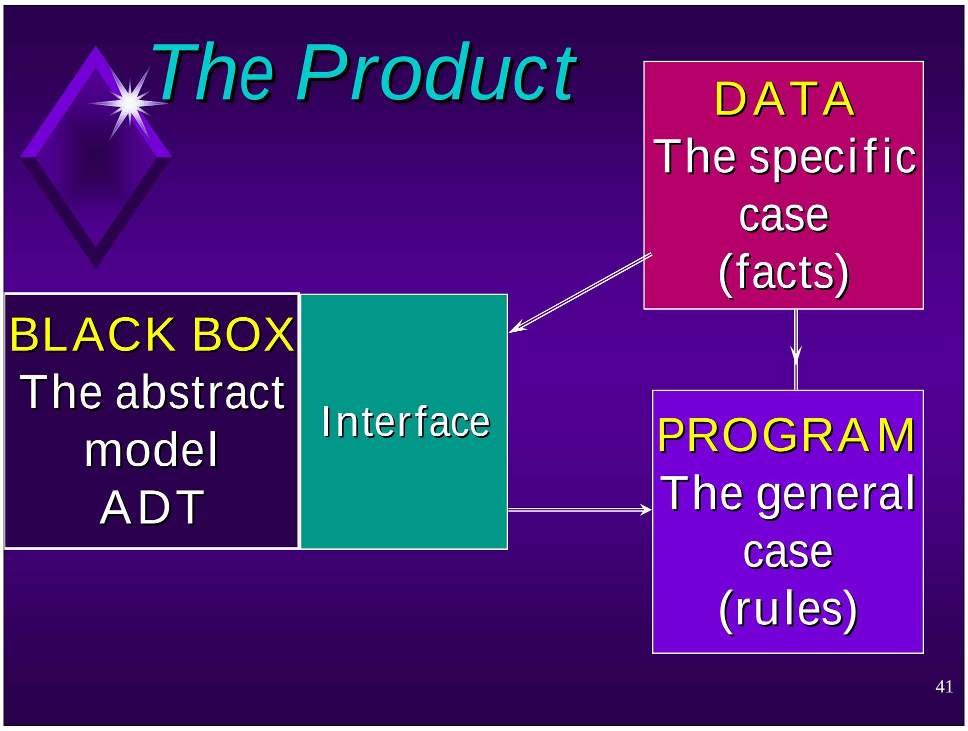
39

20



Formalization

40



The Product

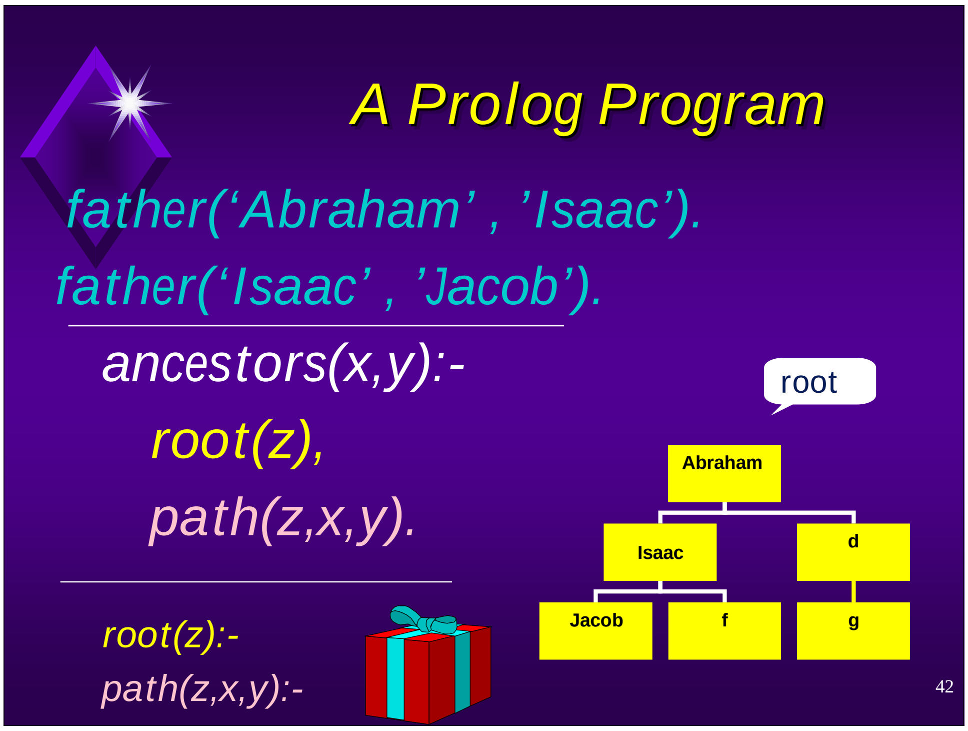
DATA
The specific case
(facts)

BLACK BOX
The abstract model
ADT

Interface

PROGRAM
The general case
(rules)

41



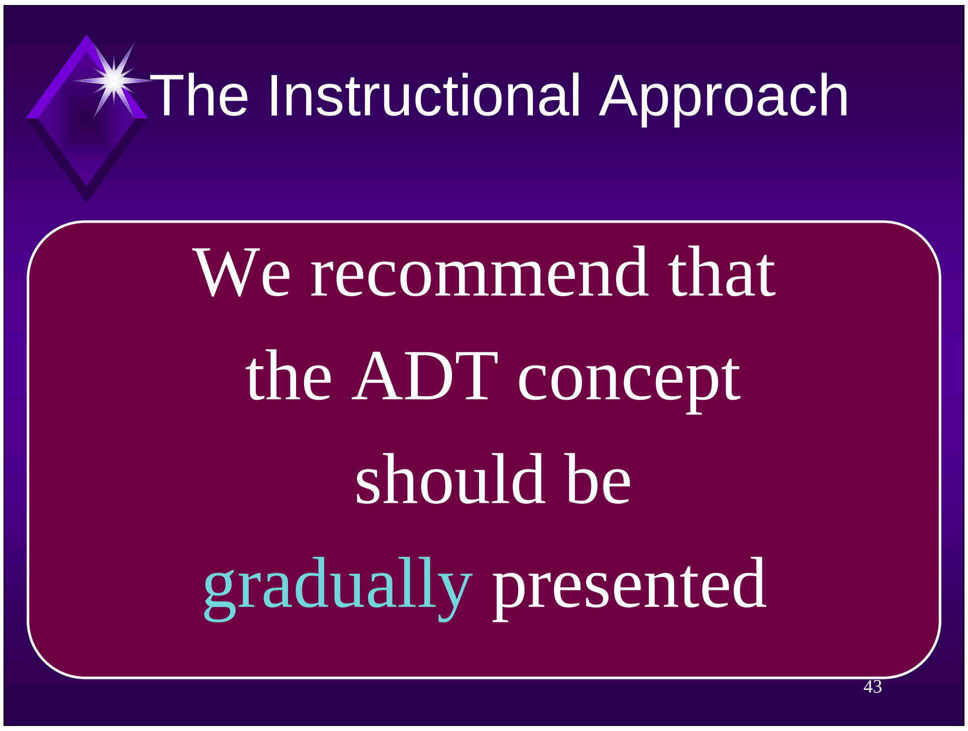
A Prolog Program

father('Abraham' , 'Isaac').
father('Isaac' , 'Jacob').

ancestors(x,y):-
root(z),
path(z,x,y).

root(z):-
path(z,x,y):-

42

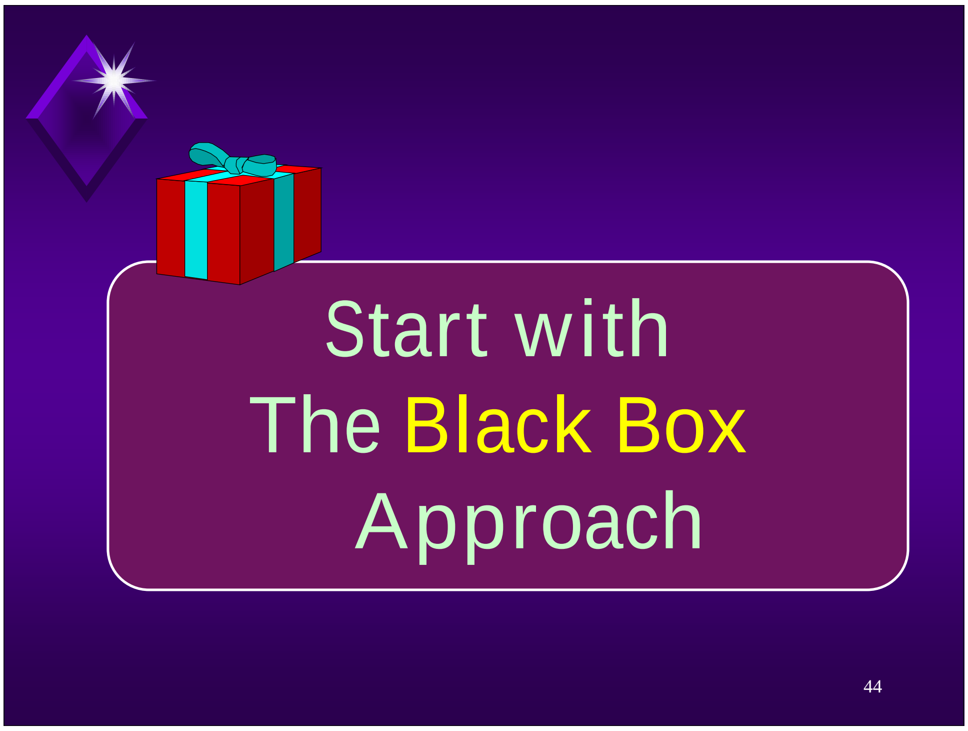


The Instructional Approach

We recommend that
the ADT concept
should be
gradually presented

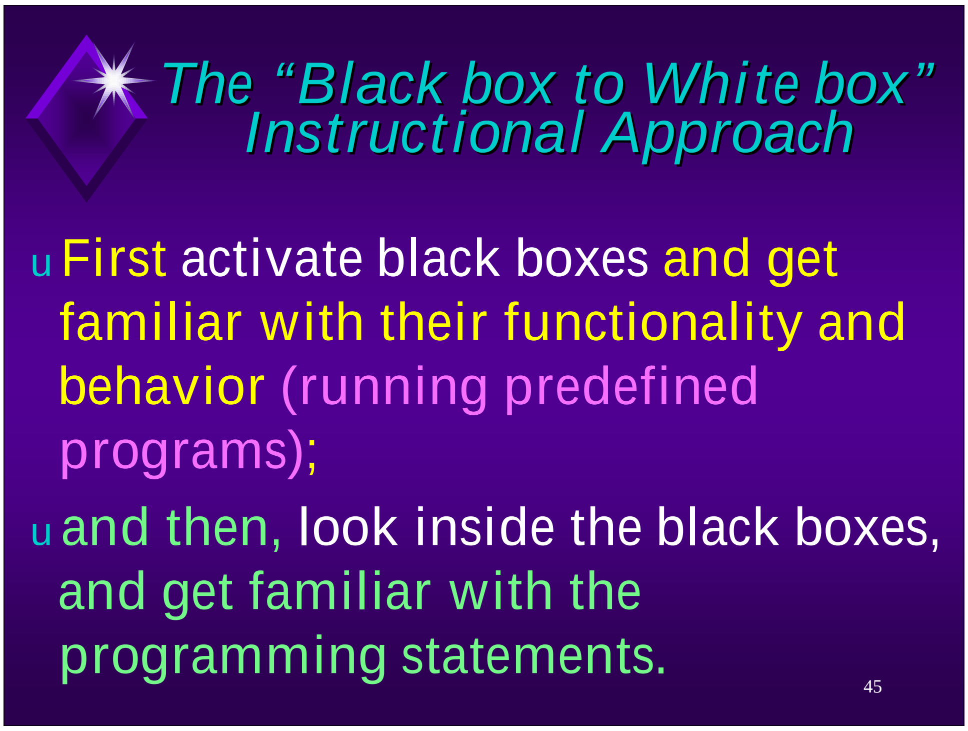
43

22



Start with
The Black Box
Approach

44



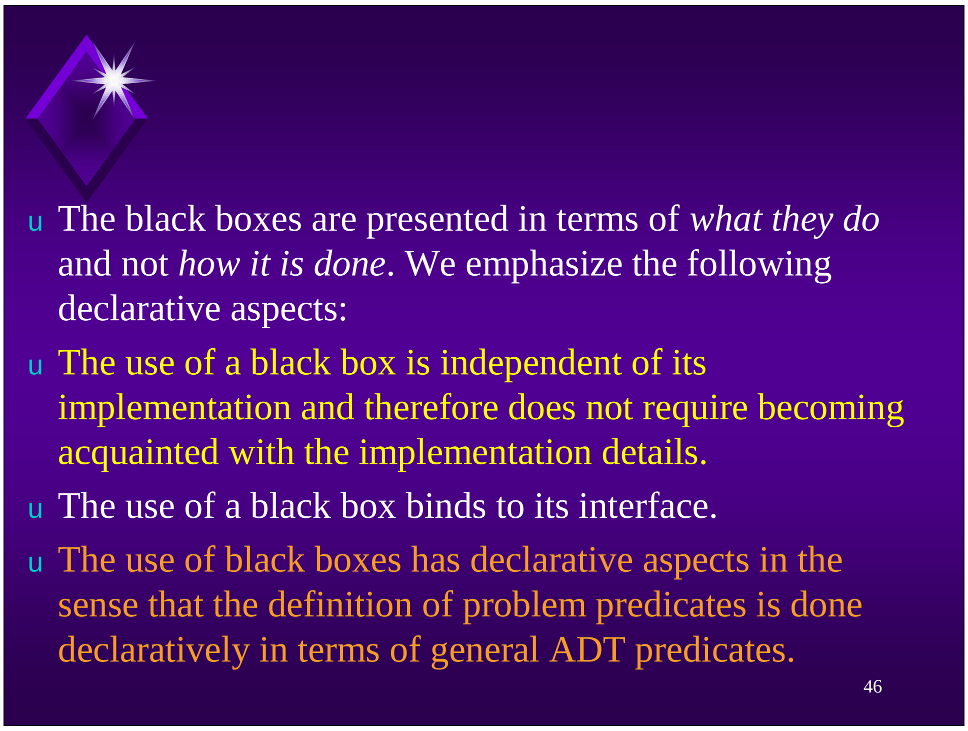
The "Black box to White box" Instructional Approach

u First activate black boxes and get familiar with their functionality and behavior (running predefined programs);

u and then, look inside the black boxes, and get familiar with the programming statements.

45

23



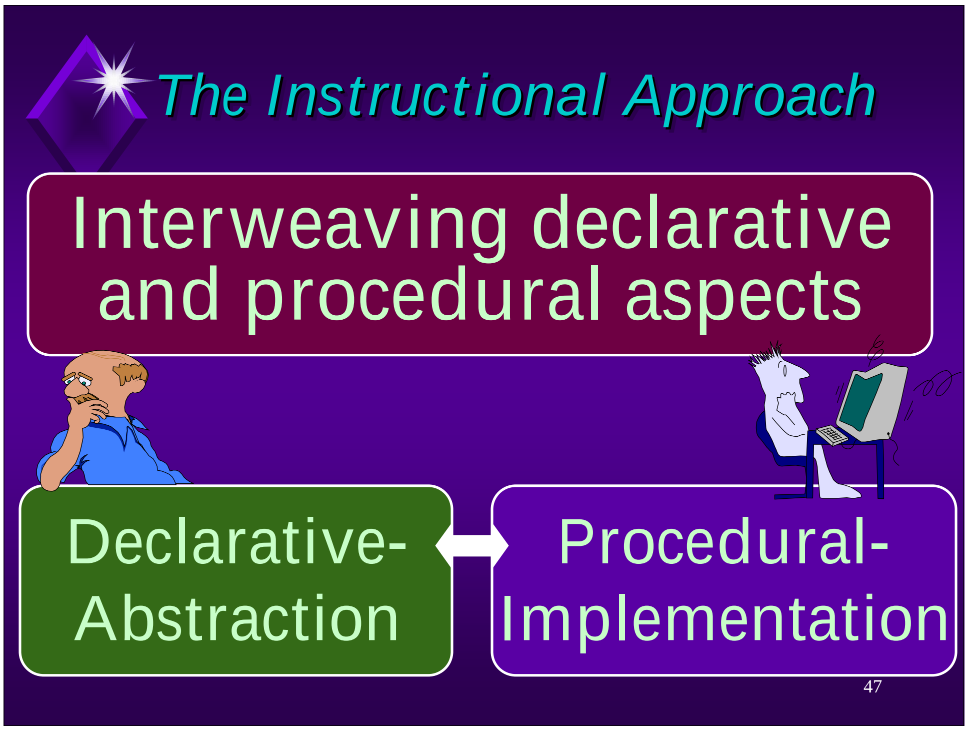
u The black boxes are presented in terms of *what they do* and not *how it is done*. We emphasize the following declarative aspects:

u The use of a black box is independent of its implementation and therefore does not require becoming acquainted with the implementation details.

u The use of a black box binds to its interface.

u The use of black boxes has declarative aspects in the sense that the definition of problem predicates is done declaratively in terms of general ADT predicates.

46

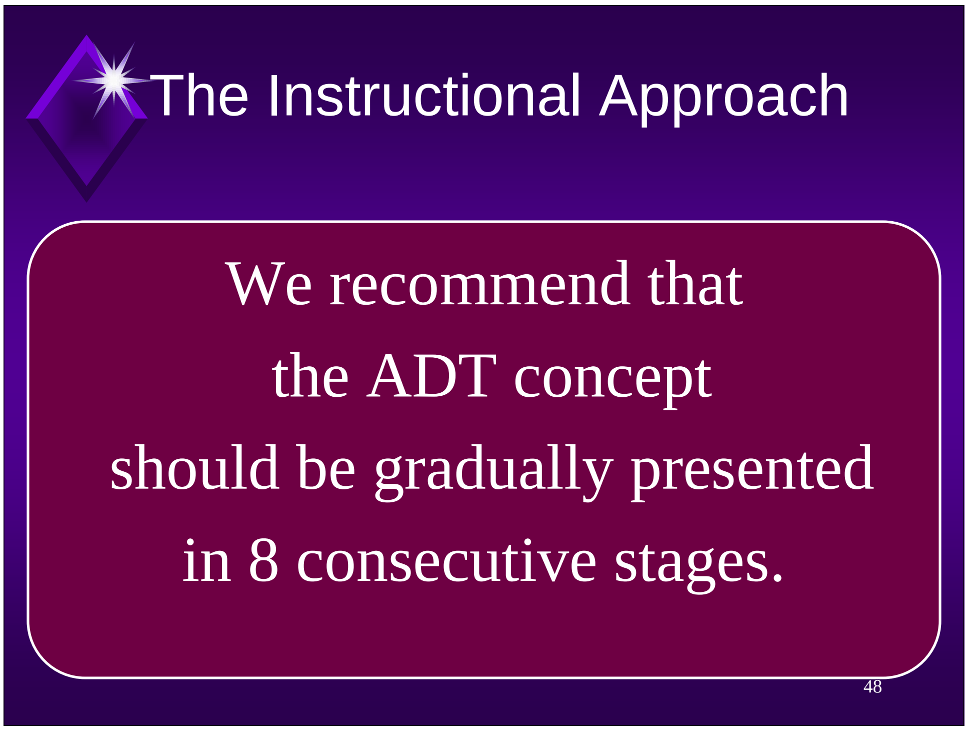


The Instructional Approach

Interweaving declarative and procedural aspects

Declarative-Abstraction ⟷ Procedural-Implementation

47



The Instructional Approach

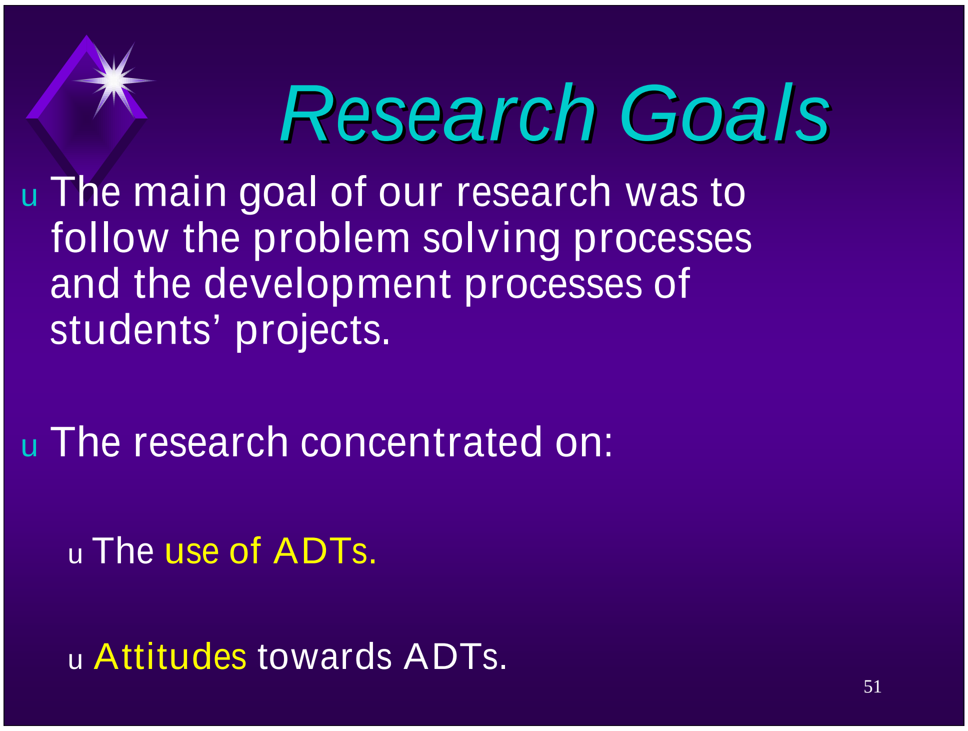
We recommend that
the ADT concept
should be gradually presented
in 8 consecutive stages.

48

	Stage	Emphasis	Target population
1	Acquaintance with given specifications of ADTs	declarative	beginners and advanced
2	Determination of ADTs to solve a given problem	declarative	
3	Use of ADT black boxes in programming	declarative and procedural	
4	Specification of new ADTs	declarative	
5	Acquaintance with ADT grey boxes	procedural	advanced only
6	Manipulation of ADT white boxes	procedural	
7	Implementation of new ADTs	procedural	
8	Knowledge integration and autonomous problem solving	declarative and procedural	

Research-

Fostering Integrative Programming Knowledge



Research Goals

u The main goal of our research was to follow the problem solving processes and the development processes of students' projects.

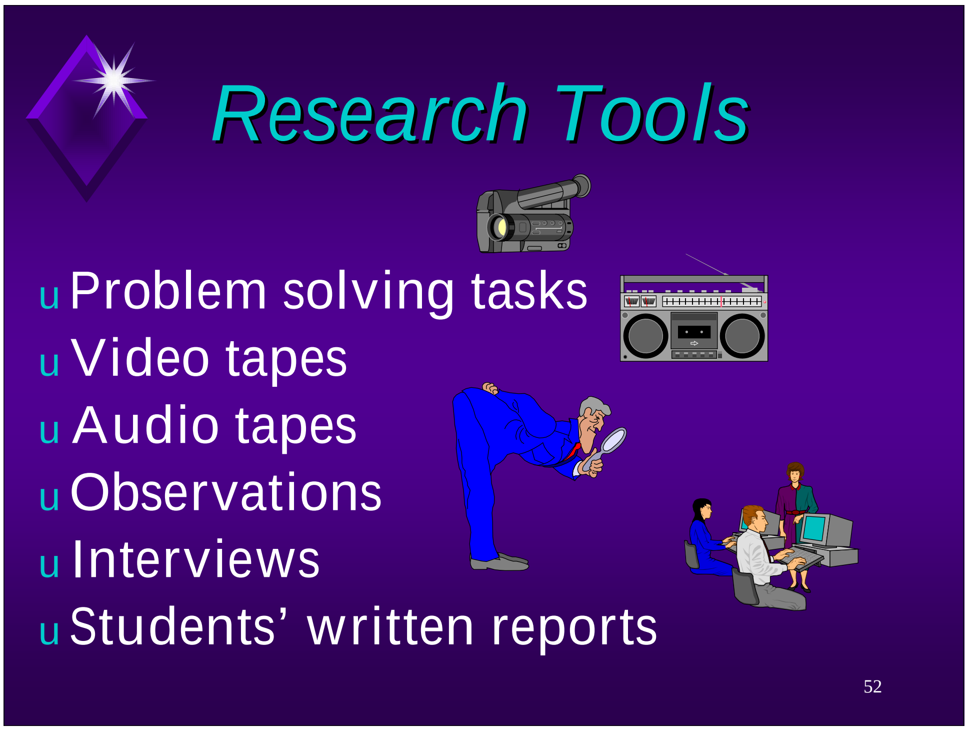
u The research concentrated on:

u The use of ADTs.

u Attitudes towards ADTs.

51

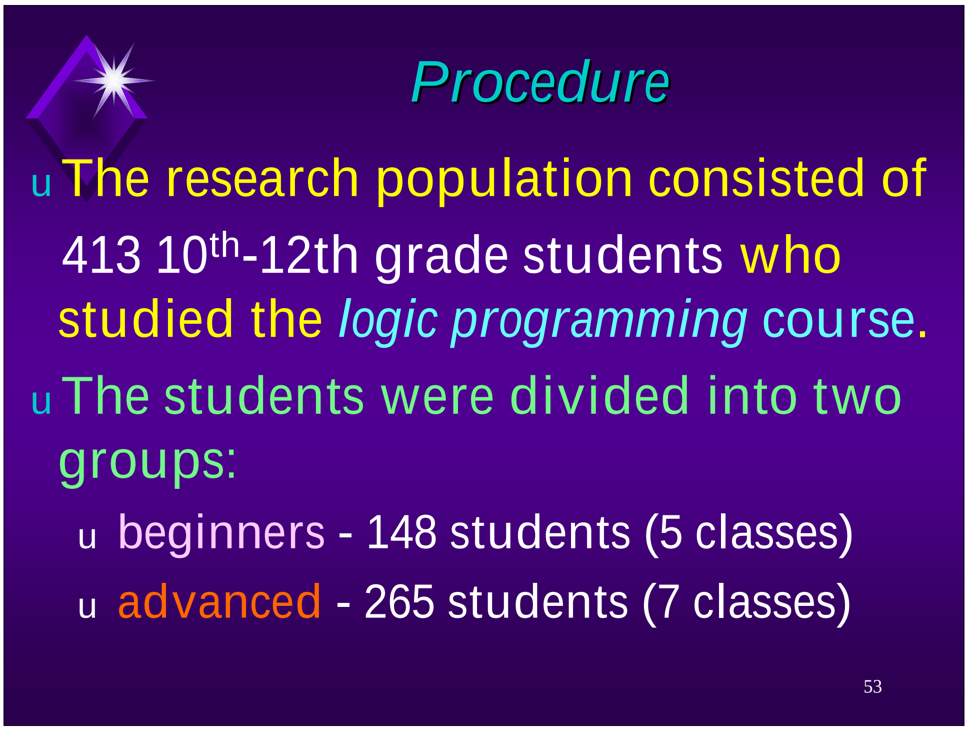
26



Research Tools

u Problem solving tasks
u Video tapes
u Audio tapes
u Observations
u Interviews
u Students' written reports

52



Procedure

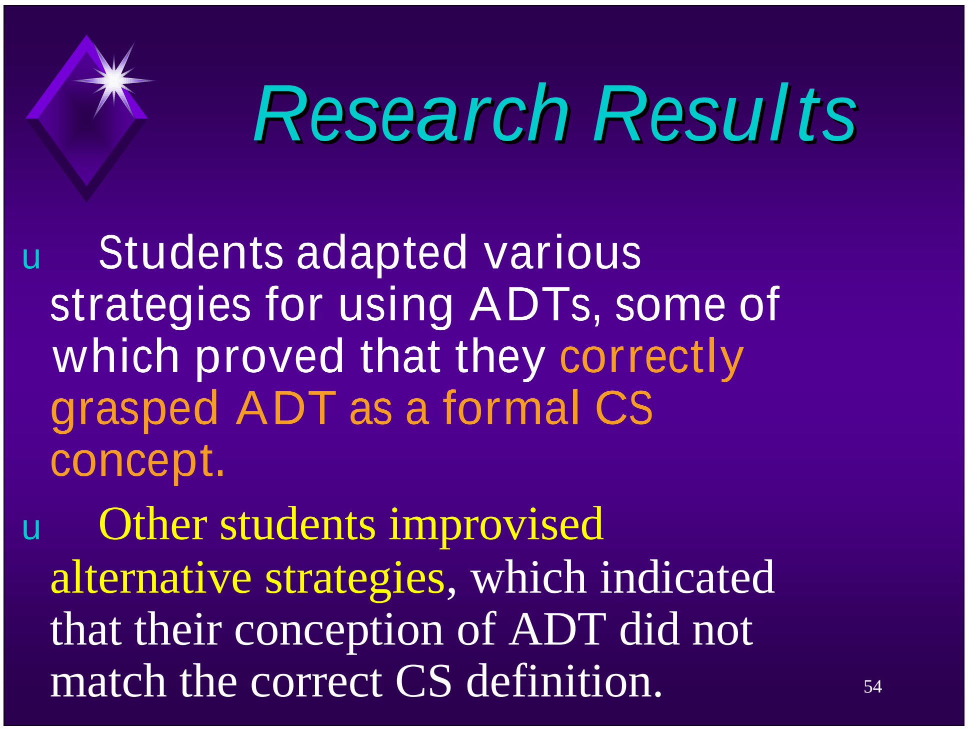
u The research population consisted of 413 10th-12th grade students who studied the *logic programming* course.

u The students were divided into two groups:

  u beginners - 148 students (5 classes)

  u advanced - 265 students (7 classes)

53

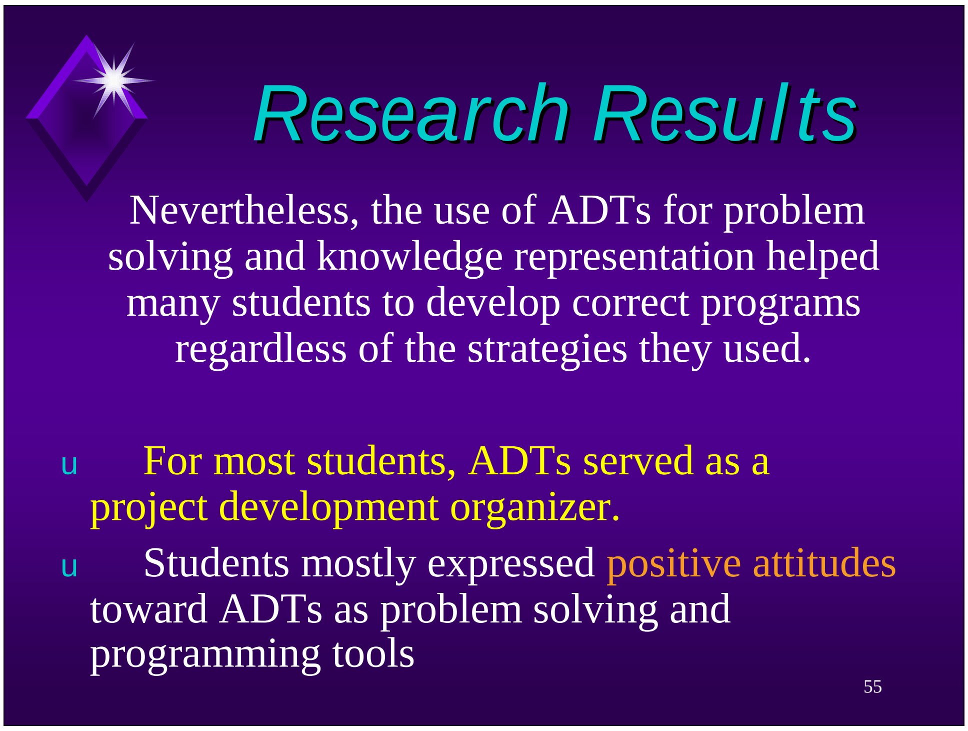


Research Results

u Students adapted various strategies for using ADTs, some of which proved that they correctly grasped ADT as a formal CS concept.

u Other students improvised alternative strategies, which indicated that their conception of ADT did not match the correct CS definition.

54

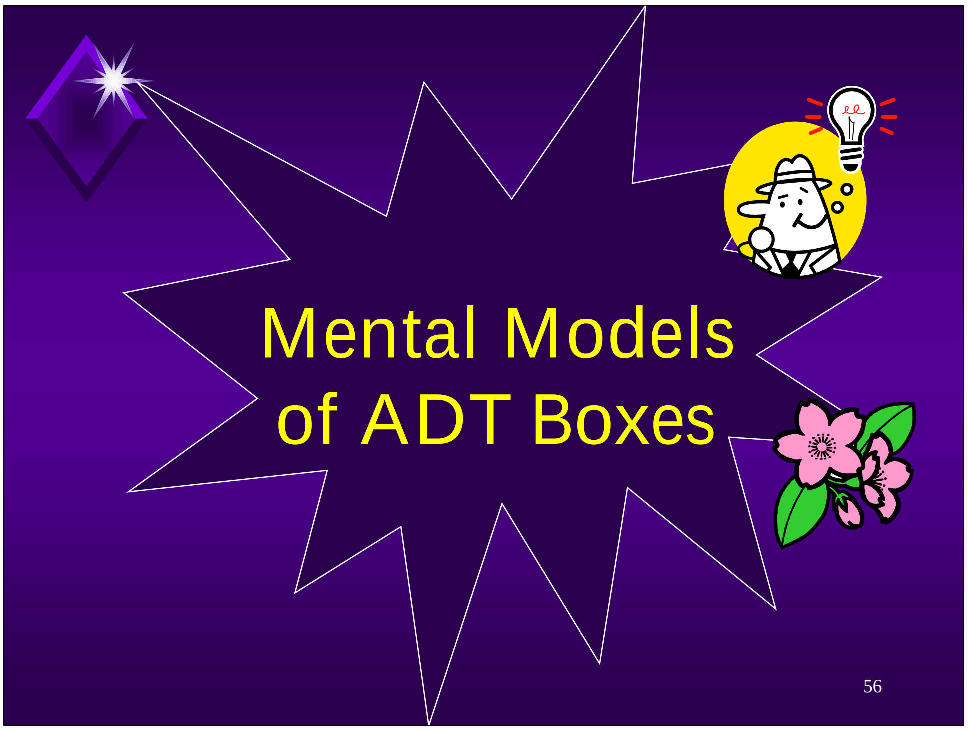


Research Results

Nevertheless, the use of ADTs for problem solving and knowledge representation helped many students to develop correct programs regardless of the strategies they used.

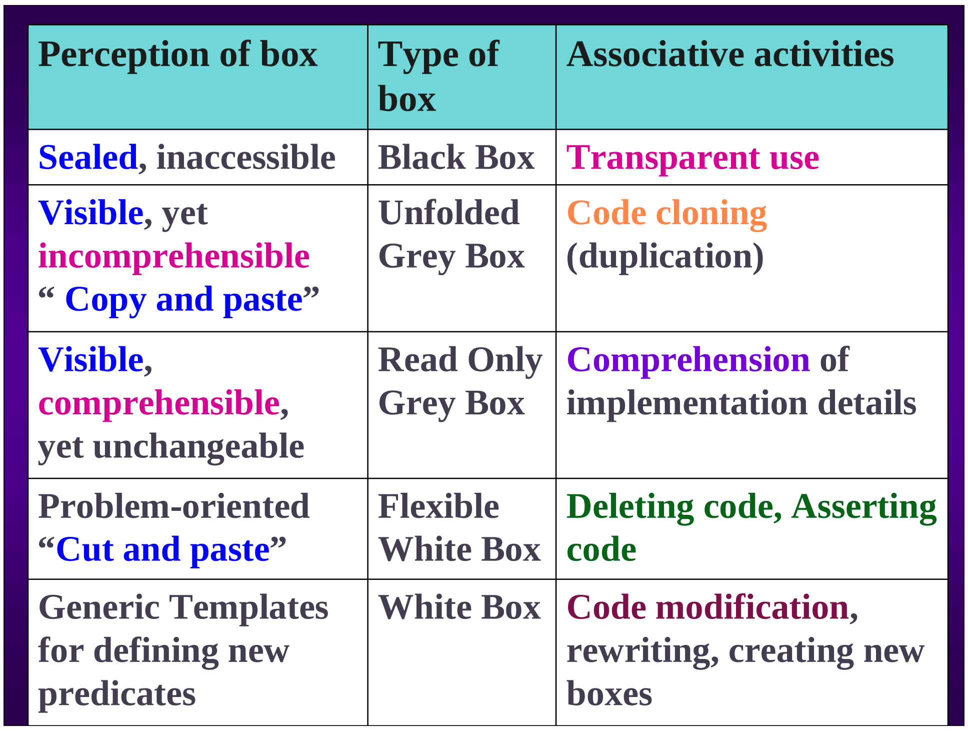
- For most students, ADTs served as a project development organizer.
- Students mostly expressed positive attitudes toward ADTs as problem solving and programming tools

55



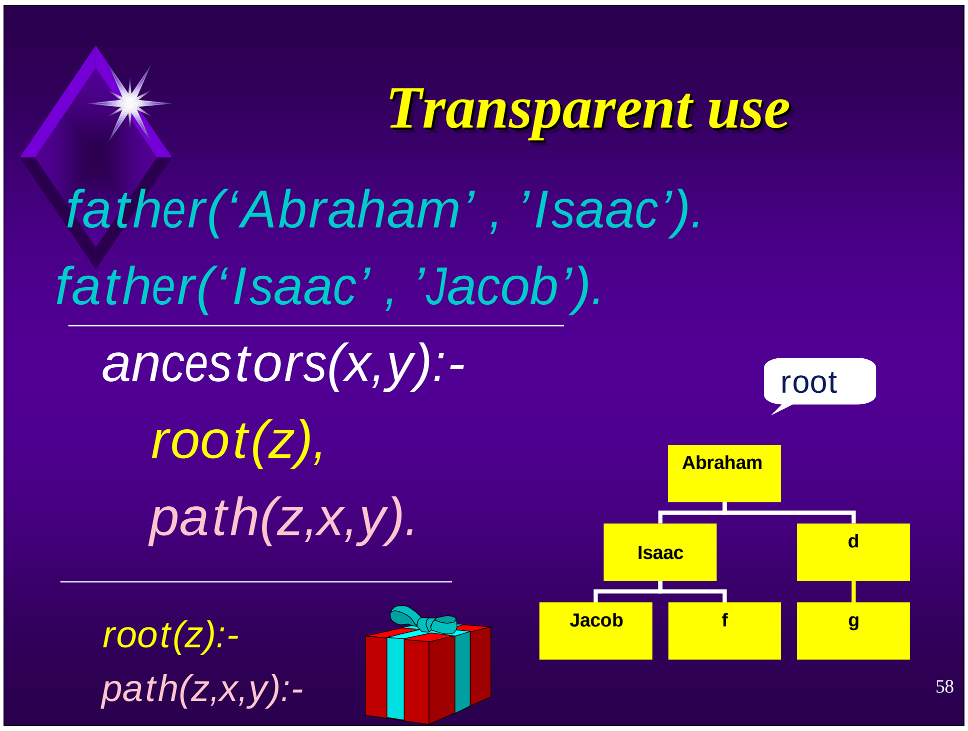
Mental Models of ADT Boxes

56



Perception of box	Type of box	Associative activities
Sealed, inaccessible	Black Box	**Transparent use**
Visible, yet **incomprehensible** “ **Copy and paste**”	Unfolded Grey Box	**Code cloning** (duplication)
Visible, **comprehensible**, yet unchangeable	Read Only Grey Box	**Comprehension** of implementation details
Problem-oriented “**Cut and paste**”	Flexible White Box	**Deleting code, Asserting code**
Generic Templates for defining new predicates	White Box	**Code modification**, rewriting, creating new boxes

29



Transparent use

father(‘Abraham’ , ’Isaac’).

father(‘Isaac’ , ’Jacob’).

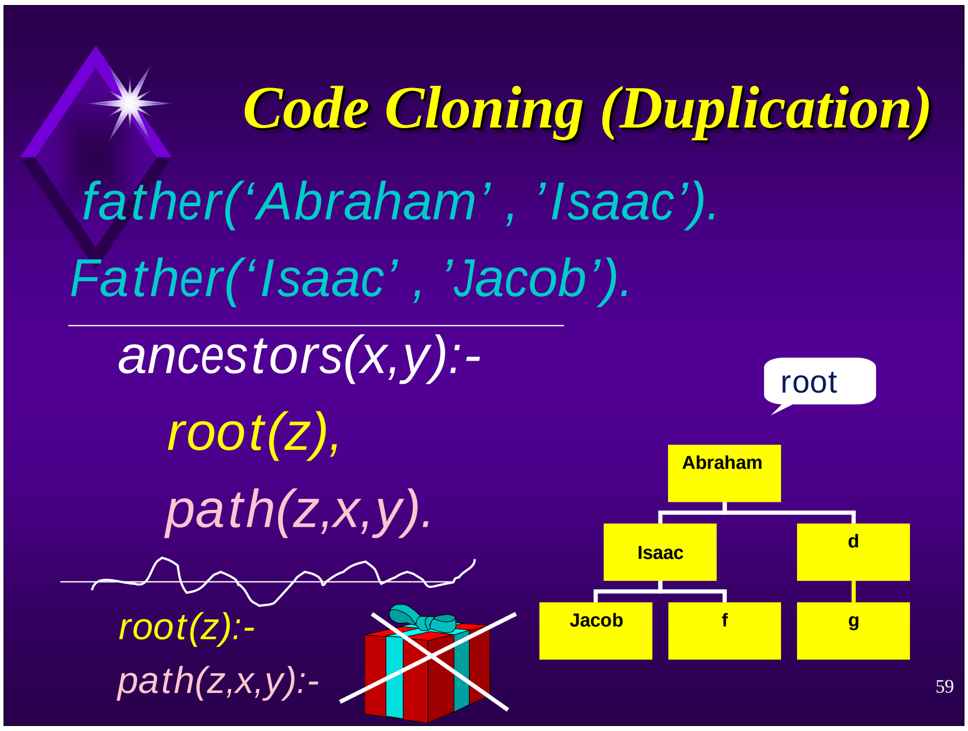
```
ancestors(x,y):-
    root(z),
    path(z,x,y).
```

root(z):-

path(z,x,y):-

root
Abraham
Isaac
d
Jacob
f
g

58

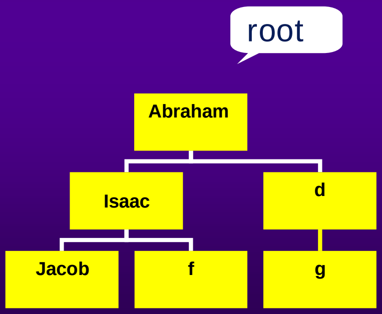


Code Cloning (Duplication)

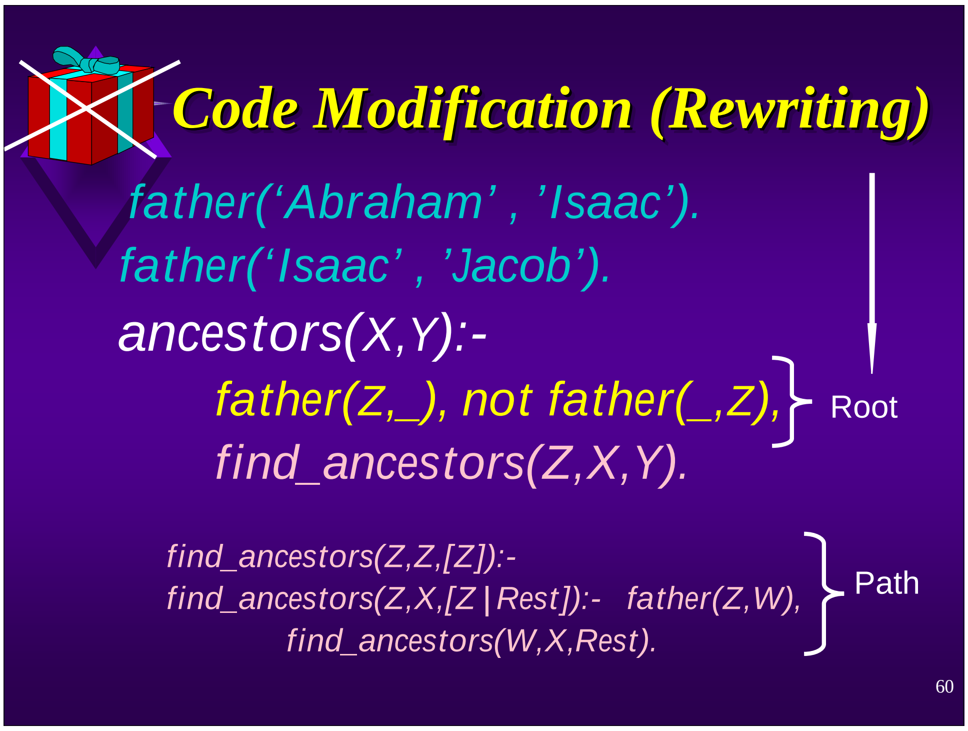
```
father('Abraham' , 'Isaac').
Father('Isaac' , 'Jacob').
```

```
ancestors(x,y):-
    root(z),
    path(z,x,y).
```

```
root(z):-
path(z,x,y):-
```



59

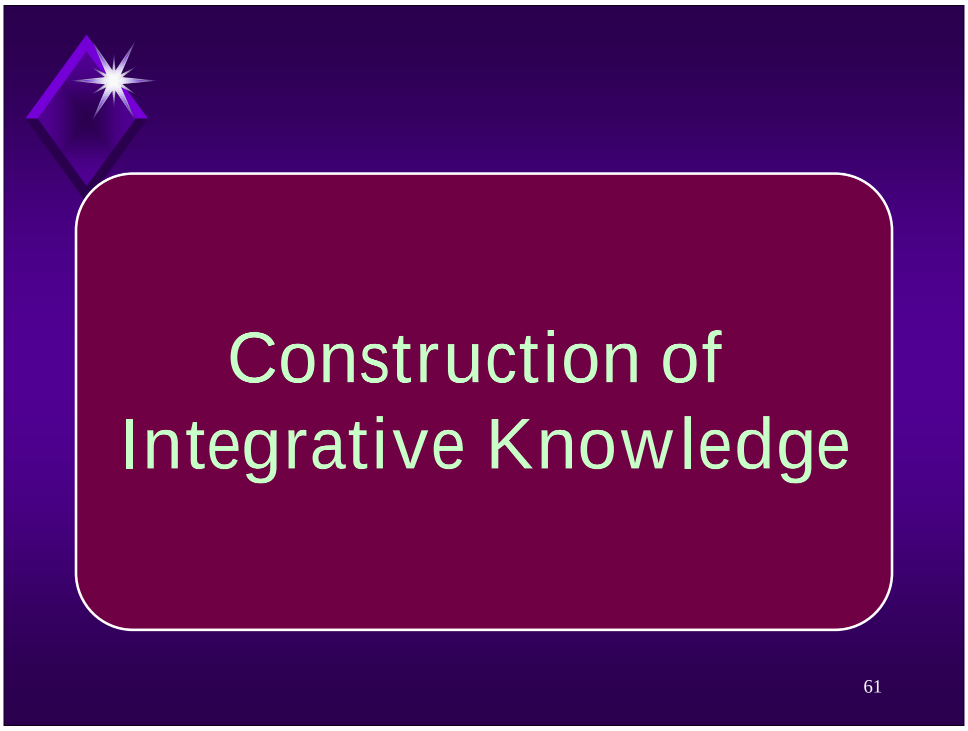


Code Modification (Rewriting)

```
father('Abraham' , 'Isaac').
father('Isaac' , 'Jacob').
ancestors(X,Y):-
      father(Z,_), not father(_,Z),   } Root
      find_ancestors(Z,X,Y).

find_ancestors(Z,Z,[Z]):-                                   } Path
find_ancestors(Z,X,[Z|Rest]):-   father(Z,W),
        find_ancestors(W,X,Rest).
```

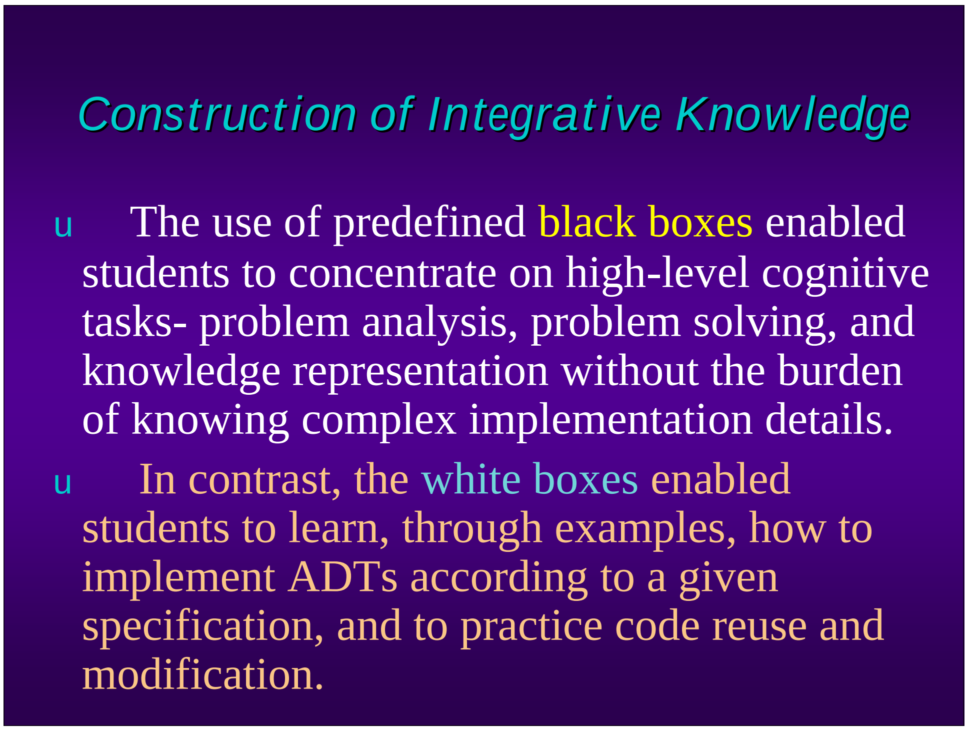
60



Construction of Integrative Knowledge

61

31



Construction of Integrative Knowledge

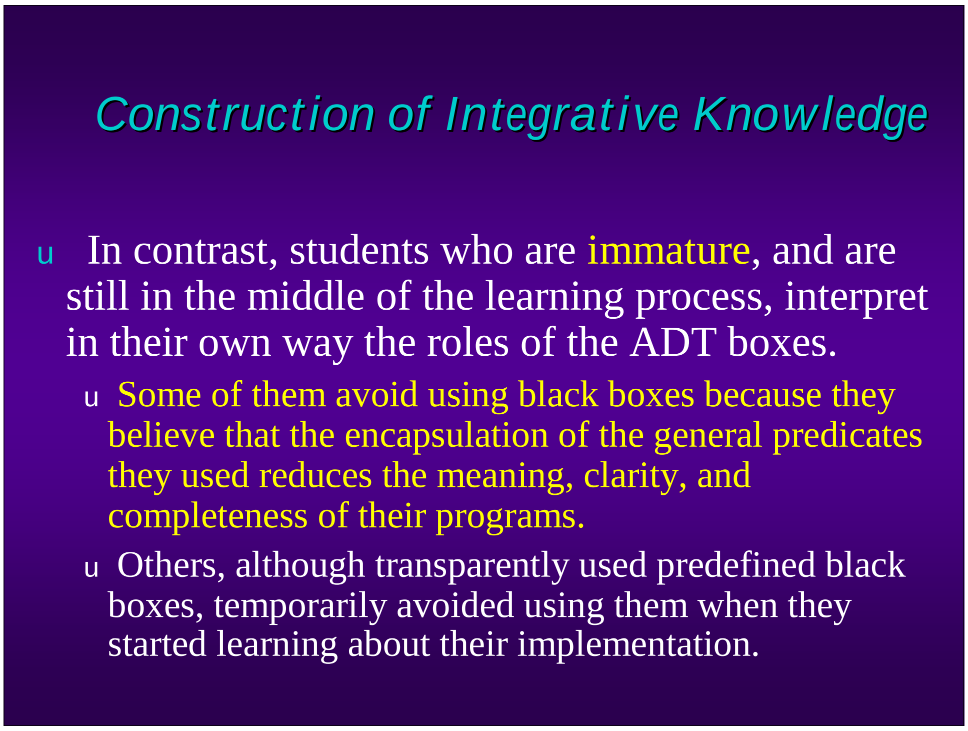
- The use of predefined black boxes enabled students to concentrate on high-level cognitive tasks- problem analysis, problem solving, and knowledge representation without the burden of knowing complex implementation details.
- In contrast, the white boxes enabled students to learn, through examples, how to implement ADTs according to a given specification, and to practice code reuse and modification.

Construction of Integrative Knowledge

The students defined their own rules of using ADT boxes and demonstrated a variety of strategies of using them while writing their programs.

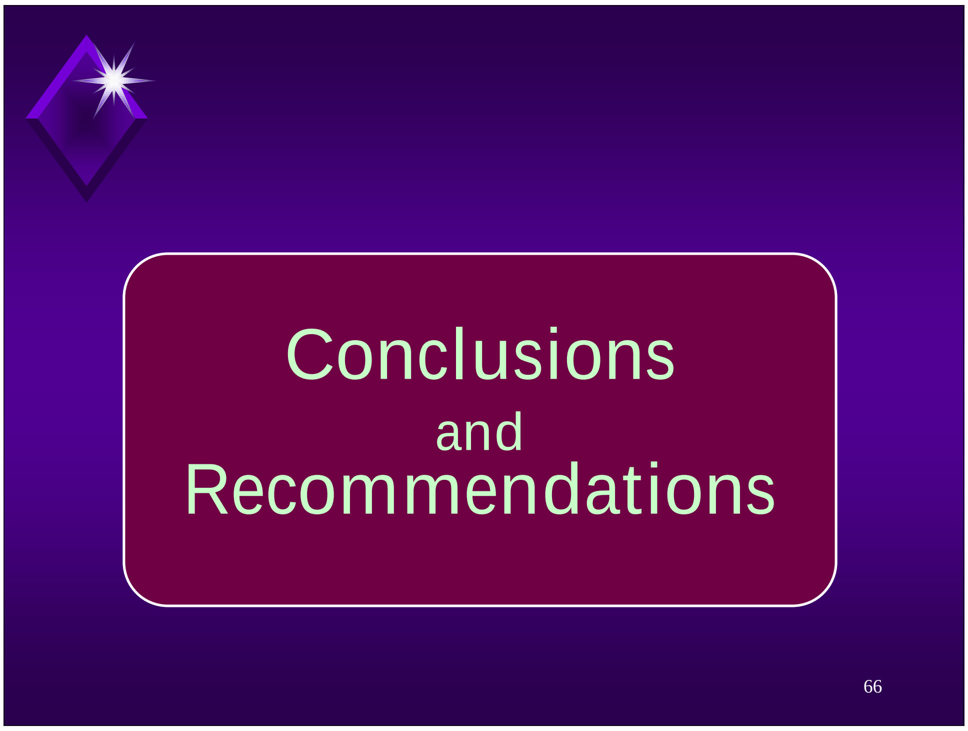
Construction of Integrative Knowledge

- Those who learned and comprehended the notions of the formal ADT concept, use it the way expert programmers do:
  - They first try to determine the suitable predefined ADT for the given problem and then transparently use the relevant ADT black box.
  - Only when the familiar predefined black boxes are insufficient to solve the problem, do they unfold a relevant box and make the minimal necessary changes, or specify and implement a new ADT.
  - Once the new ADT box is implemented, they use it transparently as is common among professionals.

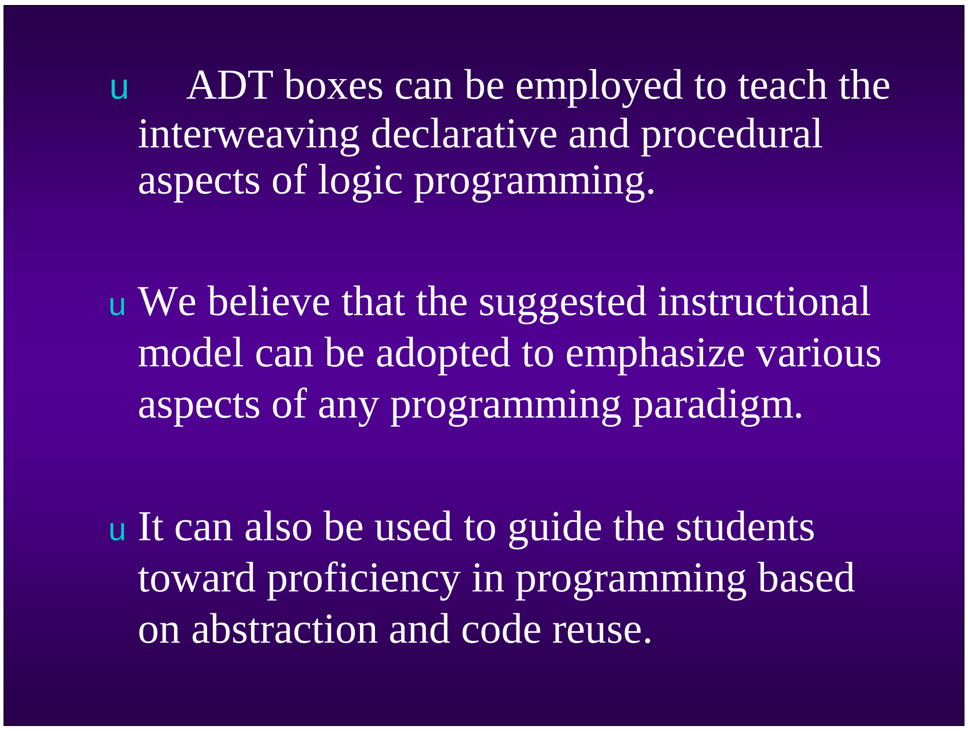


Construction of Integrative Knowledge

- In contrast, students who are immature, and are still in the middle of the learning process, interpret in their own way the roles of the ADT boxes.
  - Some of them avoid using black boxes because they believe that the encapsulation of the general predicates they used reduces the meaning, clarity, and completeness of their programs.
  - Others, although transparently used predefined black boxes, temporarily avoided using them when they started learning about their implementation.

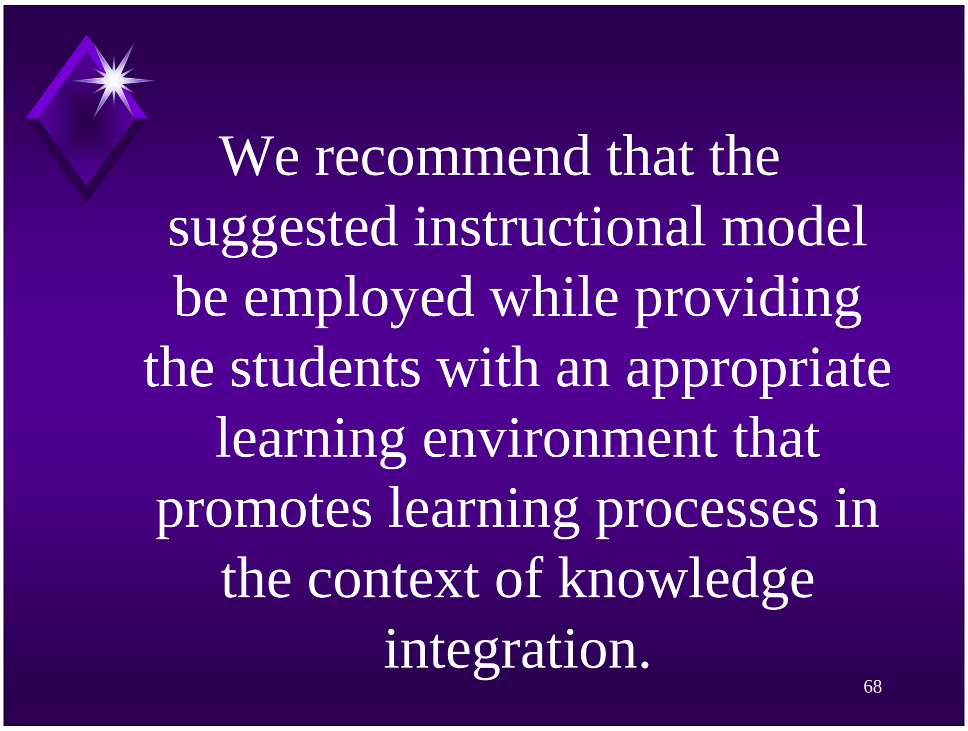


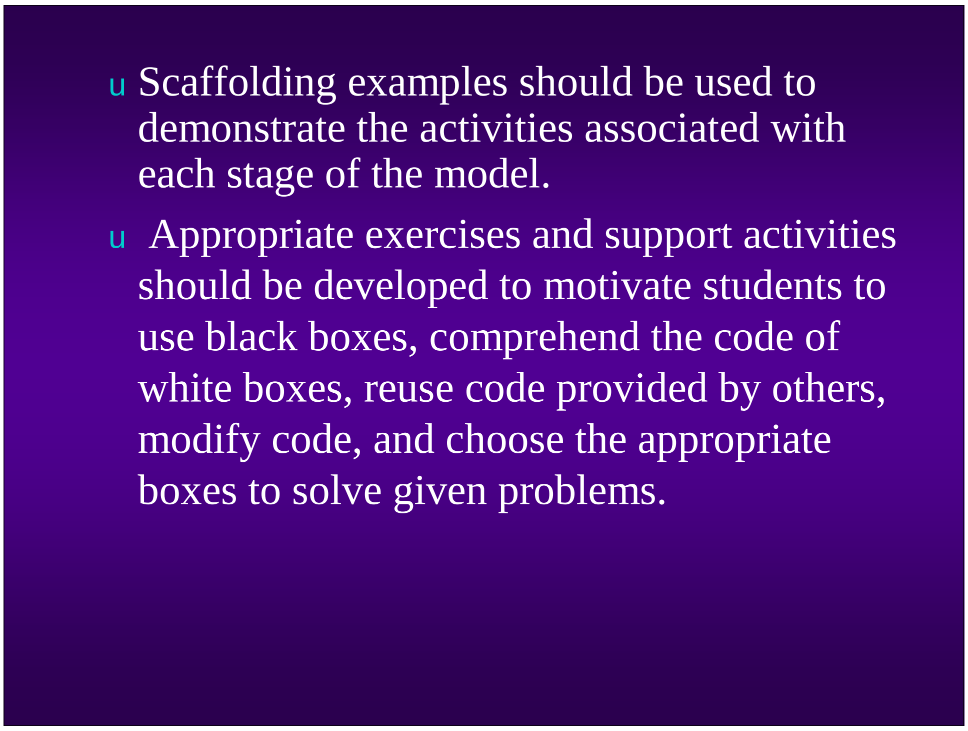
Conclusions
and
Recommendations


u ADT boxes can be employed to teach the interweaving declarative and procedural aspects of logic programming.

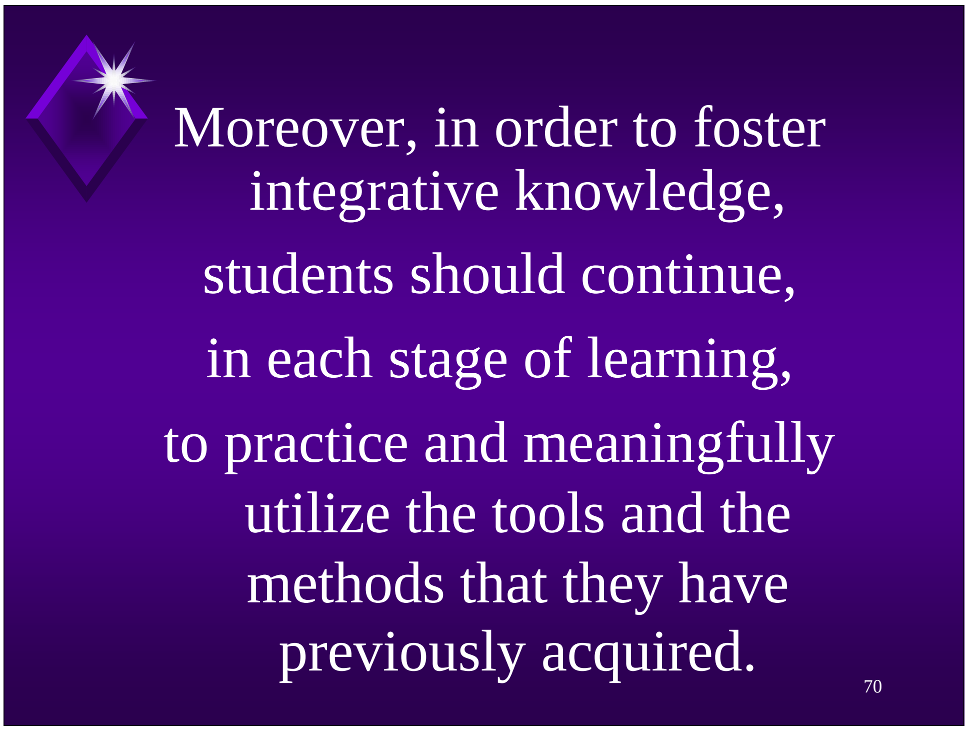
u We believe that the suggested instructional model can be adopted to emphasize various aspects of any programming paradigm.

u It can also be used to guide the students toward proficiency in programming based on abstraction and code reuse.


We recommend that the suggested instructional model be employed while providing the students with an appropriate learning environment that promotes learning processes in the context of knowledge integration.



- Scaffolding examples should be used to demonstrate the activities associated with each stage of the model.
- Appropriate exercises and support activities should be developed to motivate students to use black boxes, comprehend the code of white boxes, reuse code provided by others, modify code, and choose the appropriate boxes to solve given problems.

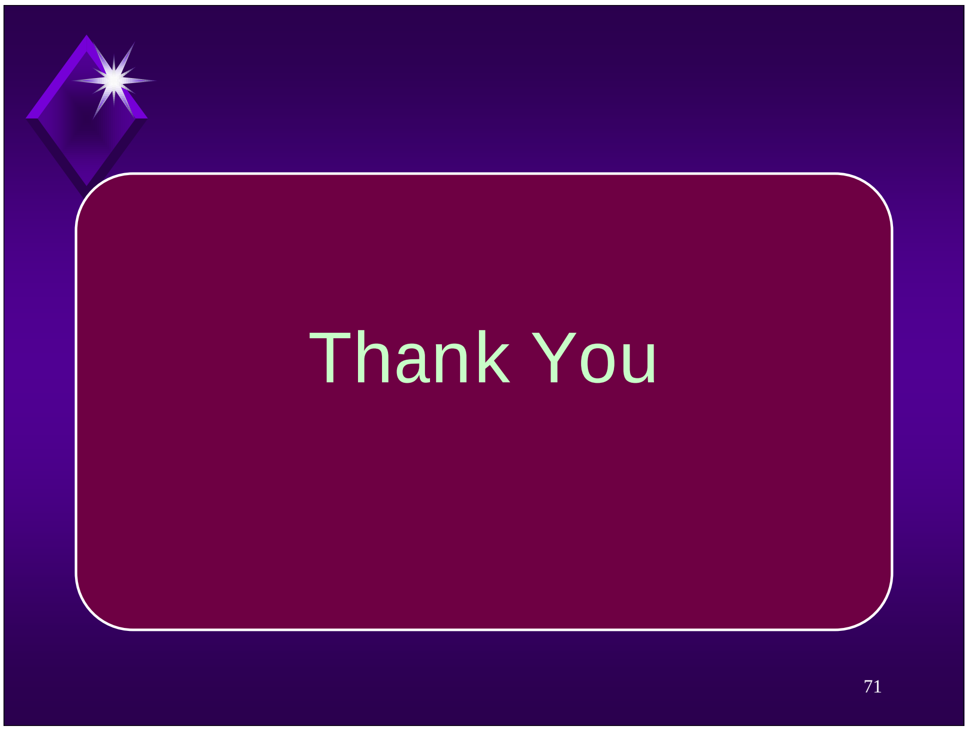


Moreover, in order to foster integrative knowledge,

students should continue,

in each stage of learning,

to practice and meaningfully utilize the tools and the methods that they have previously acquired.



Thank You

71

36