

GEM 1501 Problem Solving With Computers

Lecture 3:

Programming Languages, Data Types

Frank Stephan

Previous Lecture, Laboratory

- Laboratories

Try to learn Java Script programming 2 hours per week

All assignments which exist can be done

Java Script is one topic of examinations

No deadline for three assignments to hand in

- Previous Lecture

History of Algorithmics

Control Structures

Overview of this Lecture

- **Syntax and Semantics of Programming Languages**
- Compilers and Interpreters
- Collection of Programming Languages
- Data Types in Java Script

Programming Languages

- Need for programming languages
- Use of programming languages
- Characteristics of programming languages
- Execution of programs
- Collection of languages

Need for Programming Languages

- Programmers are humans
- Humans think in terms of abstractions
- Humans communicate with words and symbols
- Computers are machines that do very simple tasks very fast
- Programming languages bridge the gap between human programmers and computers

An Algorithm and its Automatic Translation

- **Euclid's Algorithm in German**

Gegeben seien zwei Zahlen. Solange die beiden Zahlen verschieden sind, subtrahiere man die kleinere von der grösseren und ersetze die grössere Zahl durch dieses Ergebnis. Nach dem Ende dieser Schleife haben beiden Zahlen den gleichen Wert und diesen gibt man aus.

- **Google's Translation has one error**

Two numbers are given. As long as the two numbers are different, one subtracts smaller from the larger and replaces the larger number by this result. After the end of this loop both numbers have the same value and this spend one.

- **Small Errors can spoil everything**

Programming languages have fixed and restricted syntax and a computer can understand and execute them without making any error. Every syntactically correct sentence has a precise meaning which is understandable for both, programmers and computers.

Use of Programming Languages

- Programs are precise descriptions of algorithms
- Programs are executed by computers
- Execution involves other programs (compilers, interpreters)

Characteristics of Programming Languages

- Precise syntax
- Precise semantics

Precise Syntax

- Ambiguities must be avoided:

Different syntactical notation for things which have the same name in everyday life.

- Comparison “=” and Assignment “=”

Different solutions in different languages

Java Script: `if (x==y) { z = 3; } else { z = 4; }`

```
FORTRAN 77: IF (X .EQ. Y)
              THEN Z = 3
              ELSE Z = 4 END IF
```

Pascal: `if x=y then z := 3 else z := 4`

Syntax Descriptions

- The syntactic structure of programming languages is precisely defined; for example using Backus Naur Form or Syntax diagrams
- Formal Definition, Example Java Script (simplified)

⟨Digit⟩ is: **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

⟨Number⟩ is: ⟨Digit⟩ | ⟨Number⟩ ⟨Digit⟩

⟨Name⟩ is: ⟨Letter⟩ | ⟨Name⟩ ⟨Letter⟩

⟨Expression⟩ is: ⟨Name⟩

⟨Expression⟩ is: ⟨Number⟩

⟨Expression⟩ is: ⟨Name⟩ (⟨Expression⟩ , ⟨Expression⟩)

⟨Assignment⟩ is: ⟨Name⟩ = ⟨Expression⟩

⟨Statements⟩ is: ⟨If⟩ | ⟨While⟩ | ⟨Assignment⟩ ;

⟨Statements⟩ is: ⟨Statements⟩ ⟨Statements⟩

⟨If⟩ is: **if** (⟨Boolean Condition⟩) { ⟨Statements⟩ }

⟨While⟩ is: **while** (⟨Boolean Condition⟩) { ⟨Statements⟩ }

Formal Syntax

- Backus Naur Form permits to generate programs.
- Example:
The rule “ $\langle \text{Assignment} \rangle$ is: $\langle \text{Name} \rangle = \langle \text{Expression} \rangle$ ” permits to replace “ $\langle \text{Assignment} \rangle$ ” by “ $\langle \text{Name} \rangle = \langle \text{Expression} \rangle$ ”.

- Generating a statement by successive replacement:

$\langle \text{Statements} \rangle$

$\langle \text{Assignment} \rangle$;

$\langle \text{Name} \rangle = \langle \text{Expression} \rangle$;

$\langle \text{Name} \rangle = \langle \text{Name} \rangle (\langle \text{Expression} \rangle , \langle \text{Expression} \rangle)$;

$\langle \text{Name} \rangle = \langle \text{Name} \rangle (\langle \text{Expression} \rangle , \langle \text{Number} \rangle)$;

$\langle \text{Name} \rangle = \langle \text{Name} \rangle (\langle \text{Name} \rangle , \langle \text{Number} \rangle)$;

...

$y = f(x,88888);$

Syntax Checks

- Can program be generated from its description in Backus Naur Form? – If it cannot then it has a syntactical error.

- Are all variables declared before used?

If not, the programmer forgot something or made a typing error:

```
var simulationlength;  
simulationlenthg = 23;
```

Good compilers find this type of errors.

- Names of variables and functions should avoid reserved words.

```
var while = 17; var do = true;  
do { do = f(while++); }  
    while (!do)
```

This type of programming is confusing for computers and humans.

Semantic Issues

- Programmers need to be able to precisely understand each instruction in all its different uses
- Programs need to be re-used in different environments
- Designing programming tools (e.g. debuggers)

Example: While Loops in Java Script

- Syntactic Form:

```
while ( ⟨Boolean Condition⟩ ) { ⟨Statements⟩ }
```

- Semantic Meaning:

1: Evaluate ⟨Boolean Condition⟩ .

2: If it is false, ignore ⟨Statements⟩ in loop and go to 4.

3: If it is true, do ⟨Statements⟩ and go to 1.

4: Continue by doing first command after ⟨Statements⟩ .

- Example:

```
while (x != y)
```

```
    { z = Math.max(x,y); y = Math.min(x,y); x = z-y; }
```

```
document.write("The result is "+x);
```

Example: For Loops in Java Script

- Semantics: Do loop first with $k = 0$ and last with $k = 99$ where k is each time incremented by 2.

```
for k = 0 to 99 step 2  
  do h = h+k+1  next k
```

What happens when k is never 99?

- In Java Script and C:

```
for (k=0;k<=99;k=k+2)  
  { h = h+k+1; }
```

Process clearly stops after $k \leq 99$ has become false.

Order of Operators

- Semantics tells order of operations
- Arithmetic operations - by convention, multiplication and division have priority to addition and subtraction:
 $3*4+5$ is 17 and $3+4*5$ is 23.
- Type-Conversion from numbers to string in Java Script:
`document.write("Text"+4+5);` prints "Text45"
`document.write(4+5+"Text");` prints "9Text"
Numbers converted to strings when added to strings;
Order of operators from left to right

Philosophy of Programming Languages

- Pascal: In doubt forbidden.

For example, the statement `a := "hello world"` gives a syntax error if the variable `a` is declared to be an integer number.

- Java Script: In doubt weirdly interpreted.

Automatic type conversion from numbers to text. So use brackets to make meaning clear where necessary.

```
a = "The value is "+(x+y)+".<br>";
```

Now the numbers x and y are added before being transformed into text.

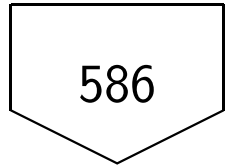
Overview of this Lecture

- Syntax and Semantics of Programming Languages
- **Compilers and Interpreters**
- Collection of Programming Languages
- Data Types in Java Script

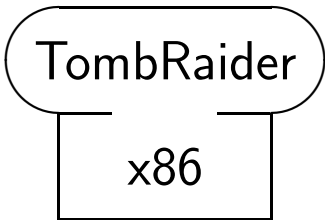
Execution of Programs

- Translators
 - Assemblers; compilers
 - Disassembler; decompilers
- Interpreters
- Combinations (virtual machines)

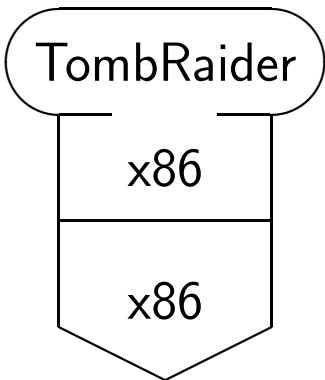
T-Diagrams



x86 Processor

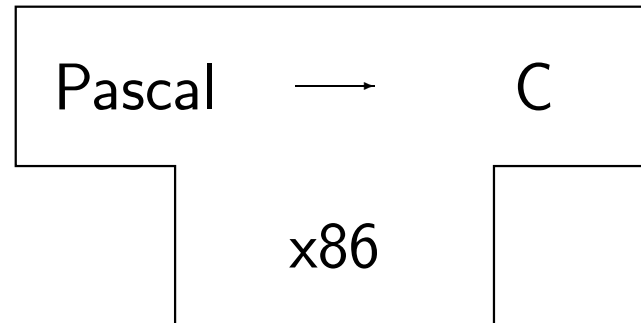


Program "TombRaider" (x86 machine code)



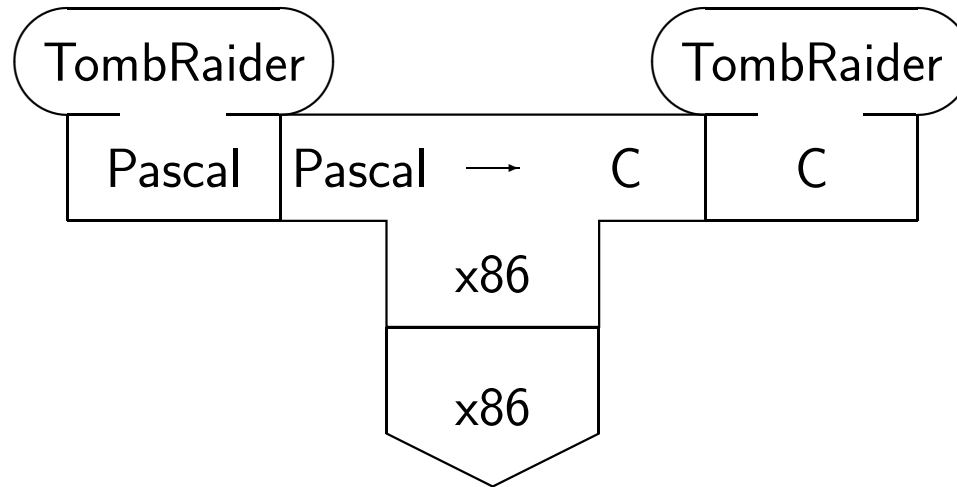
Program "TombRaider" running on x86

T-Diagram of Compiler



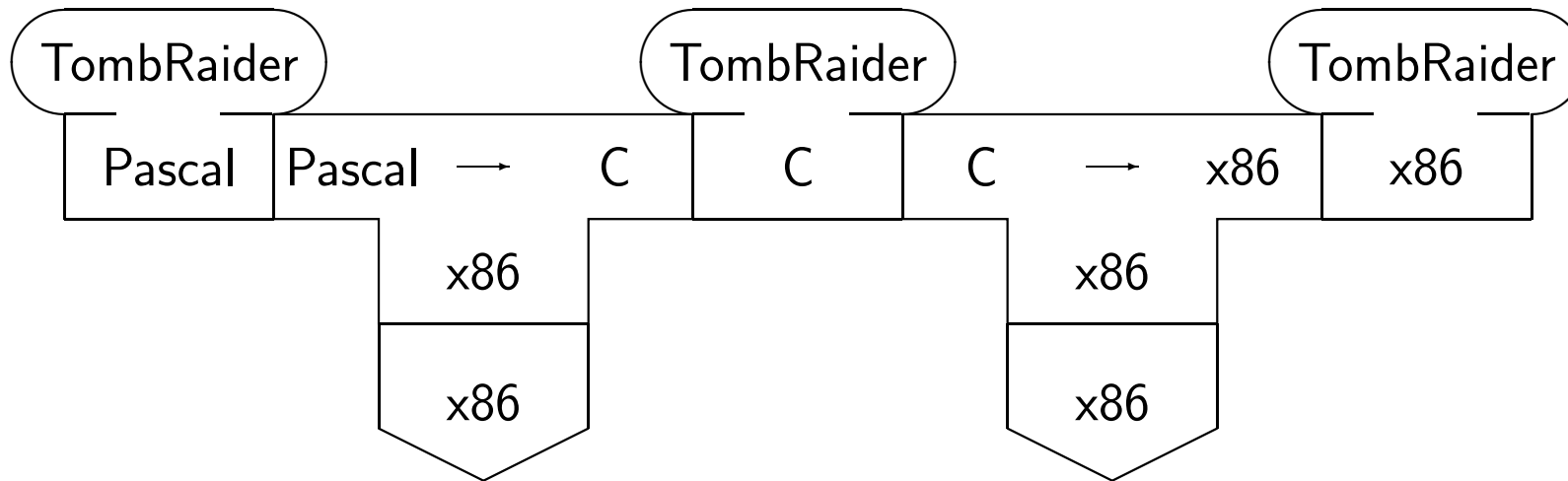
Pascal-to-C compiler in x86 machine code

Compilation



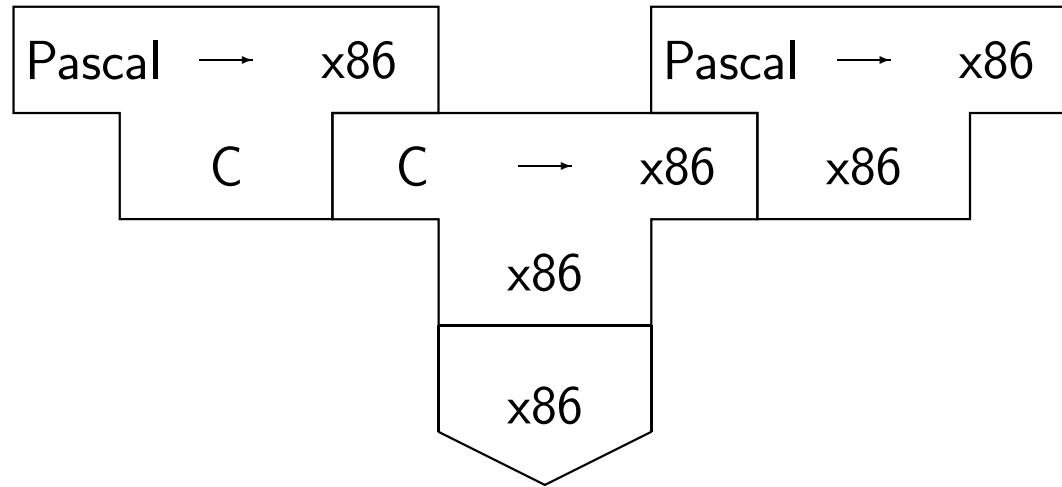
Compiling “TombRaider” from Pascal to C

Two-stage Compilation



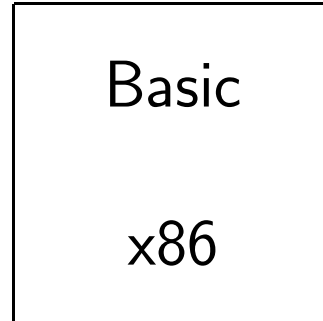
Compiling “TombRaider” from Pascal to C to x86 machine code

Compiling a Compiler



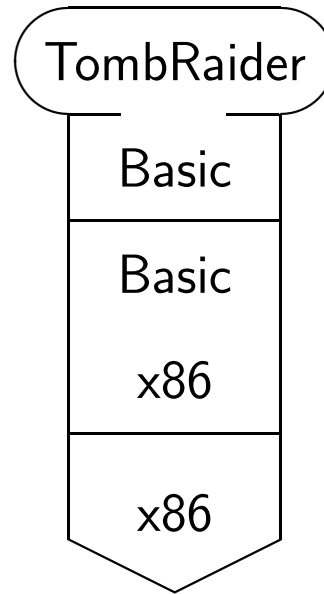
Compiling a Pascal-to-x86 compiler from C to x86 machine code

Interpreters



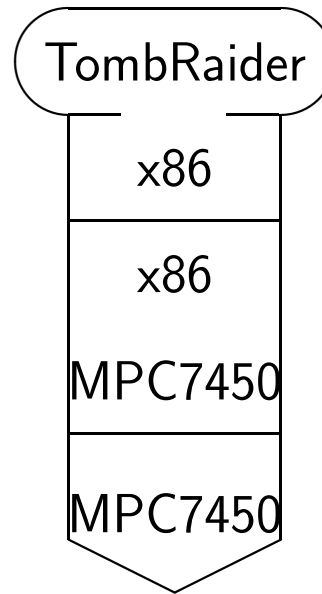
Interpreter for Basic (x86 machine code)

Interpreting a Program



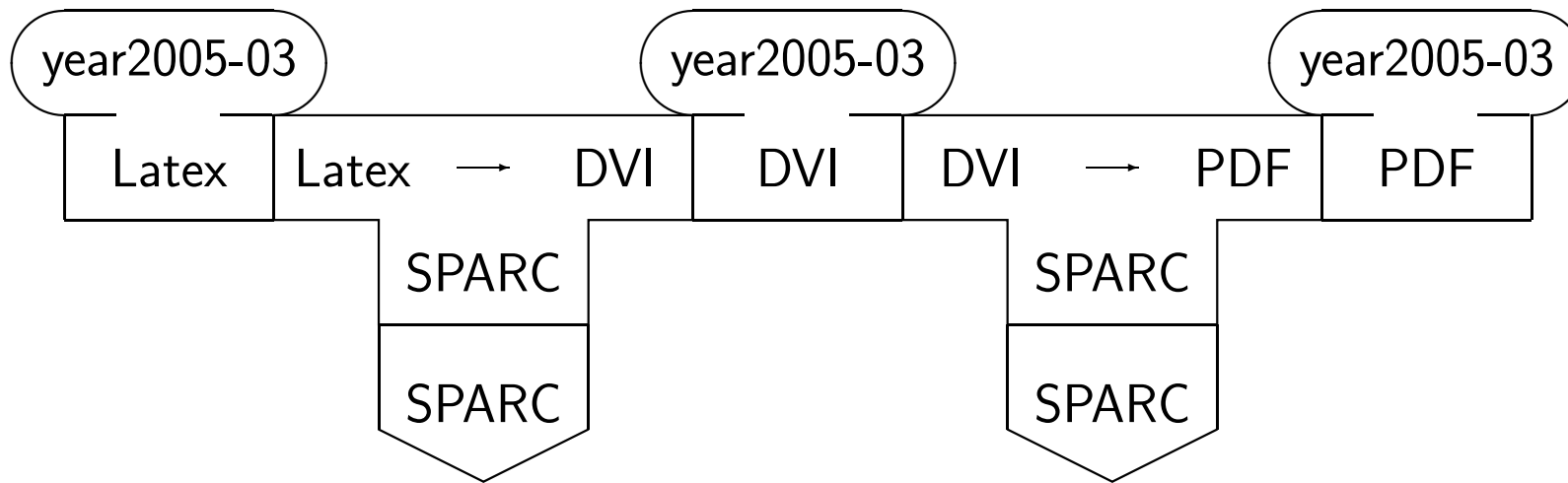
Basic program “TombRaider”
running on x86 using interpretation

Hardware Emulation



“TombRaider” x86 executable running on a PowerPC
using hardware emulation

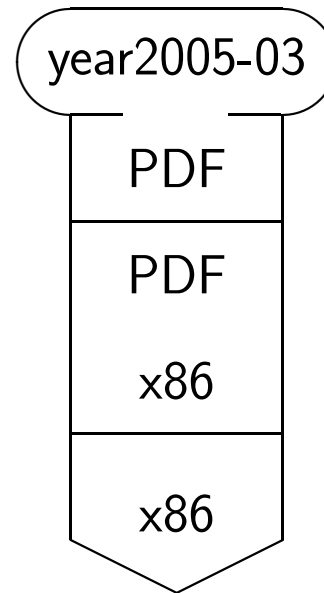
Excursion: Making these Slides



Compiling these slides

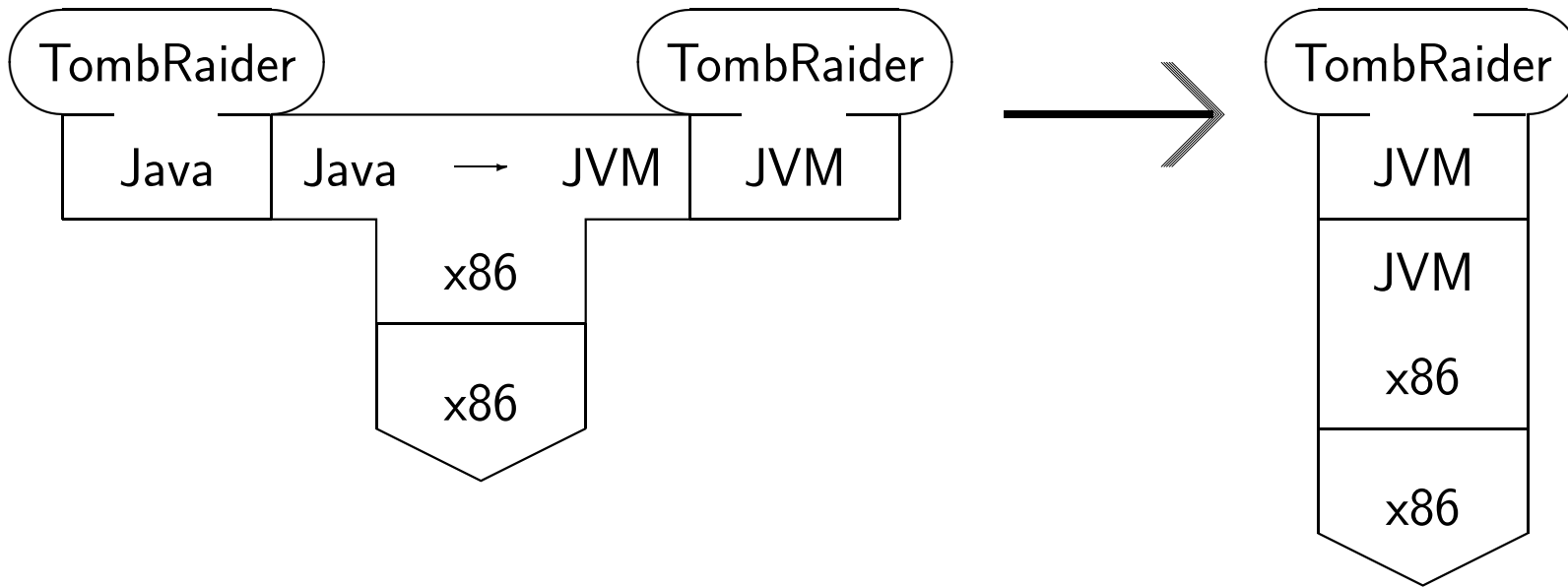
from Latex to DVI to PDF on Sun SPARC

Excursion: Viewing these Slides



Viewing the slides on this PC

Typical Execution of Java Programs



Compiling “TombRaider” from Java to JVM code, and running the JVM code on a JVM running on an x86

Summary: Execution of Programs

- Components:
programs, interpreters, compilers, machines
- T-diagrams
- Combination of interpretation and compilation is common
- Interpretation and compilation are ubiquitous in computing

Overview of this Lecture

- Syntax and Semantics of Programming Languages
- Compilers and Interpreters
- **Collection of Programming Languages**
- Data Types in Java Script

Collection of Programming Languages ‘

- Why not an algorithmic Esperanto?
- BASIC
- FORTRAN
- COBOL
- PASCAL
- C
- SNOBOL
- LISP
- PROLOG
- APL
- OZ
- JAVA SCRIPT

Why Not an Algorithmic Esperanto?

- Esperanto: Planned language, second language for everyone
Easy grammar and many borrowed words
F. Stephan: after two years learning as good as in English, much better than in Spanish (5 years learning) and Japanese (15 years learning)
Less infrastructure (no of speakers and literature) as in English
- Why not one programming language for everything?
- New developments and applications need new solutions
structured programming, object-oriented programming, parallel computing, database access, internet applications
- In Computing, new languages invented and old adapted
- New languages: clear concepts, exception-free syntax, easy to learn
- Adapted old languages: large subroutine libraries; old programs can remain in use

FORTRAN

Formula Translator

- Designed for scientific computation
- Very efficient implementations available
- Great for large array manipulations and other computationally intensive mathematical calculations
- Difficult to design data structures

COBOL

Common Business Oriented Language

- User-friendly syntax (often looks like English)

MOVE A TO B

ADD B TO C GIVING D

- Extensive support for data structures

PASCAL

Blaise Pascal (1623 - 1662), French mathematician and constructor of a mechanical calculator

- Emphasis on structured programming and clean concepts
- Type conversion has to be done explicitly
- Procedures can be defined within procedures
- Emphasis on dynamically-created data structures

Euclid's Algorithm in PASCAL

```
program euclid(input,output);  
var x,y: integer;  
begin  
  writeln('Input two positive numbers ');  
  readln(x,y);  
  while x <> y do begin  
    if x>y then x := x-y else y := y-x end;  
  writeln('Output is ',x)  
end.
```

BASIC

Beginners All-purpose Symbolic Instruction Code

- Designed for teaching purposes; became quite popular
- Originally with line numbers and Goto commands, like an improved version of Fortran
- During time, many features added and one can use it today also for Pascal-style structured programming
- Interpreted language

C

Improvement of older language B

- Machine-independent programming near system level
- Used for programming the systems UNIX and LINUX
- Many syntactic items copied by Java Script

```
main()
```

```
{ int x, y; scanf("%d %d ",&x,&y);
```

```
while (x != y)
```

```
{ if (x < y) { y = y-x; } else { x = x-y; } }
```

```
printf("The Result is %d\n",x); }
```

- Popular variant C++

SNOBOL

String Oriented Symbolic Language

- Emphasis on string and text manipulation
- Pioneering language in this area
- Today, replaced by languages such as Perl

LISP

List Processing Language

- List-oriented syntax
- Programs are naturally seen as data
- Example: (plus X Y) instead of $X + Y$
- Heavily used in Artificial Intelligence from approximately 1960 to 1990.

PROLOG

Programming in Logic

- Variables refer to (possibly partial) information
- Variables can be assigned a value only once
- Close connection to predicate logic
- Example:

```
EATS(cats,mice).
```

```
EATS(cats,frogs).
```

```
EATS(lions,cats).
```

```
EATS(X,Y) :- EATS(X,Z), EATS(Z,Y).
```

```
?- EATS(lions,frogs).
```

APL

A Programming Language

- Very concise manipulation of arrays and vectors
- Extremely rich set of operators
- Unusual syntax
- Example: `3 4 ρ⍳ 12` constructs array

```
1 2 3 4
5 6 7 8
9 10 11 12
```

OZ

Name from novel “The Wizard of Oz”

- Like PROLOG and LISP, single assignments to variables
- Combines functional, object-oriented and logic programming
- Special support for concurrent, distributed programming
- Special support for combinatorial search

JAVA SCRIPT

Java - Island in Indonesia; Script - Short interpreted language

Language for Programming Webpages

- **Many concepts relate to handling webpages**

Input and output through subwindows

Special commands for detecting type of browser

Program-code mixed into html-code of webpages

- **User of program usually not their developer**

Browsers do not display error-messages

Browsers suppress programs they detect to be faulty

- **Program runs in browser of client**

File access by reading or writing cookies on client's side

No writing of files on server's side

- **Interpreted language**

Browsers can directly execute every correct code

Overview of this Lecture

- Syntax and Semantics of Programming Languages
- Compilers and Interpreters
- Overview on Programming Languages
- **Data Types in Java Script**

Data Types in Java Script

- Standard Data Types:
Numbers: Integers and Reals;
Texts: Strings like "this is a text";
Truth values: true == 1; false == 0.
- Automatic conversion of numbers to strings.
- Variables can change data type at the assignment of a new value.

Objects

- An object is a data structure plus methods to modify the data.
- Methods can be added to object name.
- `document.write("Hello World.");` writes this phrase into the current window of the Java Script program which is called “document”.
- The mathematical object “Math” has the numbers as basic data structure and contains mathematical functions.

```
Math.sin(Math.PI)+Math.cos(Math.PI) == -1.000;  
Math.round(2.6) == 3; Math.floor(2.6) == 2; Math.ceil(2.6) == 3;  
Math.min(3,8) == 3; Math.max(3,8) == 8; Math.pow(3,8) == 6561.
```

- The methods for strings include those to determine the length and the characters of a string. If `x = "David Harel"` then `x.length == 11` and `x.charAt(4) == "d"`.

Objects

- Similar operations for different objects can have the same name:
`5+8==13` and `"5"+"8"=="58"`;
`x.length` is the length of a string `x` or the number of elements of an array `x`.
- Instances of an object can be created by a constructor function having the word “new” in front of it. For example, `y = new Array(5)` assigns to a variable with name `y` an array of 5 elements.
- Parameters of functions cannot be changed, but the components of objects can be changed.

```
function arrayupdate(a,i)
```

```
  { i = 2005;
```

not permitted

```
    a = new Array(i+1);
```

not permitted

```
    a[2005] = 23;
```

permitted

```
    return(a[i+1]); }
```

permitted

```
var x = new Array(2222); y = arrayupdate(x,32);
```

Arrays

- Several dimensions

Vector: $V[0], \dots, V[n]$

Matrix: $M[0, 0], \dots, M[0, m], \dots, M[n, 0], \dots, M[n, m]$

Tensor: more than two dimensions

- Vector of vectors: Each element $V[i]$ is a vector $V[i][0], \dots, V[i][m_i]$ itself.
- A two-dimensional Java Script array is a vector of vectors.

Multiplying $n * n$ Random Matrices

- Initializing the matrices

```
var a = new Array(n); var b = new Array(n); var c = new Array(n);
for (i=0;i<n;i=i+1)
  { a[i]= new Array(n); b[i] = new Array(n); c[i] = new Array(n);
    for (j=0;j<n;j=j+1)
      { a[i][j] = Math.random(); b[i][j] = Math.random(); } }
```

- Multiplying the matrices

```
for (i=0;i<n;i=i+1)
  { for (j=0;j<n;j=j+1)
    { c[i][j] = 0;
      for (k=0;k<n;k=k+1)
        { c[i][j] = c[i][j] + a[i][k]*b[k][j]; } } }
```

Matrices Simulated by Large Vectors

- Initializing the matrices

```
var a = new Array(n*n); var b = new Array(n*n);
var c = new Array(n*n);
for (i=0;i<n;i=i+1)
  { for (j=0;j<n;j=j+1)
    { a[i*n+j] = Math.random(); b[i*n+j] = Math.random(); } }
```

- Multiplying the matrices

```
for (i=0;i<n;i=i+1)
  { for (j=0;j<n;j=j+1)
    { c[i*n+j] = 0;
      for (k=0;k<n;k=k+1)
        { c[i*n+j] = c[i*n+j] + a[i*n+k]*b[k*n+j]; } } }
```

Records

- A record is a variable consisting of several parts.
- Each part contains an entry which can be a number, a string or another composed variable.
- Records are realized as objects in Java Script
- Components are accessed with names after the dot
- Example of record “owner” from salary addition

```
owner.fullname == "Eberhard Ei";  
owner.salary == 0.00;  
In short, owner == ("Eberhard Ei",0.00);
```

Eberhard Ei owns the company but works elsewhere.

Data of Salary Example

- Array of Records
- Constructor Function creates object with two entries
keyword “this” refers to just created object

```
function employee(fn,s1)
  { this.fullname = fn;
    this.salary = s1; }
```

- Each entry in array is created by calling constructor function

```
allemployees = new Array(4);
allemployees[0] = new employee("Anneliese Aal",1324.55);
allemployees[1] = new employee("Boris Bratling",3322.11);
allemployees[2] = new employee("Claudia Creme",4000.00);
allemployees[3] = new employee("Doris Dattel",1010.10);
```

Adding the Salaries

- The adding part of the program

```
function salarysum()  
  { var notednumber = 0.00; var ind;  
    for (ind=0;ind<allemployees.length;ind=ind+1)  
      { notednumber = notednumber+allemployees[ind].salary; }  
    return(notednumber) }  
var total = salarysum();  
document.write("Sum of salaries is SGD "+total+"<br>");
```

- In the loop, the variable `ind` has as upper bound the length of the array which is obtainable as a method of the array object.
- function `salarysum()` accesses the data structure `allemployees` as a global variable and not as a parameter.
- <http://www.comp.nus.edu.sg/~gem1501/salary.html>

Stacks

- Push objects onto the stack; pop object from the stack; test whether a stack is empty.
- In Java Script implemented with Arrays.

```
var fruits = new Array("oranges","apples");  
fruits.push("pears","grapes");  
var poppedfruit = fruits.pop();
```

- After these operations

```
fruits == ("oranges","apples","pears");  
poppedfruit == "grapes";
```

- The length of the stack is `fruits.length`, thus one can test emptiness.

Queues

- Push element after the last element of the queue; take element from the first position of queue; test whether a queue is empty.
- In Java Script implemented with Arrays.

```
var fruits = new Array("oranges", "apples");  
fruits.push("pears", "grapes");  
var firstfruit = fruits.shift();
```

- After these operations

```
fruits == ("apples", "pears", "grapes");  
firstfruit == "oranges";
```

- The method `fruits.unshift("bananas");` pushes an element from the front. The outcome is `("bananas", "apples", "pears", "grapes")`

Databases

- Large quantity of information in tables
phonebooks, railway timetables, passenger information systems
- Approaches: hierarchical databases and relational databases
Special programming languages for retrieving and changing data
SQL: Structured Query Language
Programs mixture of normal commands and SQL commands
- Internet could be viewed as some type of large data base
Search engines: almost no syntax needed
Users place data on the internet
Search engines produce auxiliary databases for finding fast
- Data warehouses: databases which do not change fast
- Data mining: Automatic learning from data in stored databases
historic documents, DNA sequences, astronomical data, internet data

Next Week

Algorithmic methods

- Searches and traversals
- Divide and conquer
- Greedy algorithms
- Dynamic programming