Overview of Introduction

- Abstraction: language versus linguistics, abstraction, complexity, visualization, and expression
- Paradigms and Aspects
- Ancient History of Written Language
- Development of Mathematical Language
- Early History of Programming Languages
Abstraction

Let’s think about what *abstraction* means. When we learn about the world around us, we all make mistakes. For example, we call palm trees by that name because they are tall plants, even though botanists tell us that they are related to grasses and not to trees.

For another example, consider when we learn English we learn the pattern of the past tense of verbs:

- play → played
- work → worked
- move → moved
- open → opened

But what is the past tense of “to go,” or “to run?”
Other examples of *analogy* in English linguistics . . . What is the plural of the (obscure) words *ziff*, *zo*, and *zax*? Why, *ziffs*, *zos*, and *zaxes*, of course. The required plurals are formed according to a pattern already familiar from a large number of other English nouns. Consider the plural of a small class of nouns derived from Latin:

- cactus: cacti
- hippopotamus: hippopotami
- radius: radii
- succubus: succubi
- syllabus: syllabi

What is the plural of *octopus*?
AN ACCIDENTAL BITE BY A RADIOACTIVE SPIDER HAS GIVEN TEENAGER PETER PARKER SUPER POWERS, AND TRANSFORMED HIM INTO --

the AMAZING SPIDER-MAN

SPIDERS ARE SMALL AND FAST, DOC...

OCTOPUSES ARE BIG AND CLUMSY.

YOU'RE WRONG ABOUT THAT!

I KNOW, I SHOULD HAVE SAID "OCTOPUS!"
Many people would say *octopi*, but because the word is of Greek origin, its Greek plural is *octopodes*. The problem is three competing plural patterns: English, Latin, Greek. Fowler’s Modern English Usage states that “the only acceptable plural is English is *octopuses*,” and that *octopi* is misconceived and *octopodes* is pedantic. *Octopi* derives from the mistaken notion that *octōpūs* is a second declension Latin noun, which it is not. It is originally from the ancient Greek masculine noun *oktopous* (οκτώπους), whose plural is *oktopodes* (οκτωποδες). Examples from Trask, *Historical Linguistics*, page 106.

In this class we are pedantic, or we prefer to say careful and precise. We are precise for two reasons: for the sake of scientific accuracy, and because computers are stupid and unforgiving.
How many have studied a foreign language?
<table>
<thead>
<tr>
<th><strong>English</strong></th>
<th><strong>German</strong></th>
<th><strong>Spanish</strong></th>
<th><strong>Turkish</strong></th>
<th><strong>Hungarian</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Datei</td>
<td>Fichero</td>
<td>Dosya</td>
<td>Állomány</td>
</tr>
<tr>
<td>New</td>
<td>Neu</td>
<td>Nuevo</td>
<td>Yeni</td>
<td>Új</td>
</tr>
<tr>
<td>Open</td>
<td>Öffnen</td>
<td>Abrir</td>
<td>Aç</td>
<td>Megnyitás</td>
</tr>
<tr>
<td>Close</td>
<td>Schließen</td>
<td>Cerrar</td>
<td>Kapat</td>
<td>Bezárás</td>
</tr>
<tr>
<td>Save</td>
<td>Speichern</td>
<td>Salvar</td>
<td>Kaydet</td>
<td>Mentés</td>
</tr>
<tr>
<td>Print</td>
<td>Drucken</td>
<td>Imprimir</td>
<td>Yazdir</td>
<td>Nyomtatás</td>
</tr>
<tr>
<td>Exit</td>
<td>Beenden</td>
<td>Salir</td>
<td>Çıkış</td>
<td>Kilépés</td>
</tr>
<tr>
<td>Edit</td>
<td>Bearbeiten</td>
<td>Editar</td>
<td>Değiştir</td>
<td>Szerkesztés</td>
</tr>
<tr>
<td>Undo</td>
<td>Rückgängig</td>
<td>Anular</td>
<td>Geri Al</td>
<td>Vissza</td>
</tr>
<tr>
<td>Redo</td>
<td>Wiederholen</td>
<td>Rehacer</td>
<td>Tekrarla</td>
<td>Újra</td>
</tr>
<tr>
<td>Cut</td>
<td>Ausschneiden</td>
<td>Cortar</td>
<td>Kes</td>
<td>Kivág</td>
</tr>
<tr>
<td>Copy</td>
<td>Kopieren</td>
<td>Copiar</td>
<td>Kopyala</td>
<td>Másol</td>
</tr>
<tr>
<td>Paste</td>
<td>Einfügen</td>
<td>Pegar</td>
<td>Yapıştıır</td>
<td>Beilleszt</td>
</tr>
</tbody>
</table>
What is linguistics?
What is linguistics?

In linguistics, language is the subject, not how to speak a language. Likewise, in the study of programming languages, it is the medium (programming language) that is the subject, not how to program in some language. To quote an aphorism by Marshall McLuhan (1911–1980), patron saint of Wired magazine: “the Medium Is the Message.” Usually it is important to write programs to accomplish some task. Often in this class, we are not interested program’s task, but rather the medium in which it is written.
Linguistics: the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is abstraction.
Linguistics: the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is abstraction.

Concepts in linguistics: grammar, word order (SVO, etc.), i.a.

Case:
Linguistics: the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is abstraction. Concepts in linguistics: grammar, word order (SVO, etc.), i.a.

Case: indicates the main role of a construct in a sentence, e.g., dative case (object), genitive case (possessive).

Predicate:
**Linguistics**

**Linguistics:** the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is *abstraction.*

Concepts in linguistics: grammar, word order (SVO, etc.), i.a.

**Case:** indicates the main role of a construct in a sentence, e.g., dative case (object), genitive case (possessive).

**Predicate:** the part of a sentence that expresses what is said of the subject and usually consists of a verb with or without objects.

**Gender:**
**Linguistics**

**Linguistics:** the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is *abstraction*.

Concepts in linguistics: grammar, word order (SVO, etc.), i.a.

**Case:** indicates the main role of a construct in a sentence, e.g., dative case (object), genitive case (possessive).

**Predicate:** the part of a sentence that expresses what is said of the subject and usually consists of a verb with or without objects.

**Gender:** a class of words in a language that may be arbitrary or based on distinguishable characteristics (as shape, social rank, or sex) and determines agreement with and selection of other words.

**Agglutination:**
Linguistics: the study of the units, nature, structure, and modification of language. Not a language, the difference between the study of a language and linguistics is abstraction. Concepts in linguistics: grammar, word order (SVO, etc.), i.a. 

Case: indicates the main role of a construct in a sentence, e.g., dative case (object), genitive case (possessive).

Predicate: the part of a sentence that expresses what is said of the subject and usually consists of a verb with or without objects.

Gender: a class of words in a language that may be arbitrary or based on distinguishable characteristics (as shape, social rank, or sex) and determines agreement with and selection of other words.

Agglutination: the formation of words by putting together constituents of which each contributes a definite meaning.
Abstraction

“There is a general distinction concerning thinking: that between categories and individuals, or classes and instances.” page 351. Thus something as simple as a newspaper might be specified to six levels, as in Douglas R. Hofstadter’s illustration of that ambiguity, with a progression from abstract to concrete in *Gdel, Escher, Bach* (1979):

1. a publication
2. a newspaper
3. The San Francisco Chronicle
4. the May 18 edition of the Chronicle
5. my copy of the May 18 edition of the Chronicle
6. my copy of the May 18 edition of the Chronicle as it was when I first picked it up (as contrasted with my copy as it was a few days later: in my fireplace, burning)
Abstraction

Abstraction

**Abstraction:** conceptualization without reference to specific instances. The act of determining the fundamental, essential, important, intrinsic properties disassociated from the specific details. A model or representation of something including only the essential properties.

*Software engineering is all about abstraction. Every single concept, construct, and method is entirely abstract. Of course, it doesn’t feel this way to most software engineers. But that’s my point. The main benefit they got from the mathematics they learned in academia was the experience of rigorous reasoning with purely abstract objects and structures.*


Abstraction is the key to computing as it is to mathematics. Both disciplines ask the questions: what is the essential essence of stuff?

In computing we are bound by the immutable rules of construction.
As expressed by Benjamin C. Pierce in *Types and Programming Languages* (2002), the abstraction principle reads:

*Each significant piece of functionality in a program should be implemented in just one place in the source code. Where similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts.*

A key challenge in programming language design is providing mechanisms to abstract out all conceivable commonality.
Don’t Repeat Yourself (DRY) or Duplication is Evil (DIE) is a principle of software development aimed at reducing repetition of information of all kinds. The DRY principle is stated as “Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.” The principle has been formulated by Andy Hunt and Dave Thomas in their book *The Pragmatic Programmer*. They apply it quite broadly to include “database schemas, test plans, the build system, even documentation.”
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6</td>
<td>...</td>
</tr>
<tr>
<td>2 4 6 8 10 12</td>
<td>...</td>
</tr>
<tr>
<td>4 9 14 19 24 29</td>
<td>...</td>
</tr>
<tr>
<td>1 4 9 16 25 36</td>
<td>...</td>
</tr>
<tr>
<td>1 3 7 15 31 63</td>
<td>...</td>
</tr>
<tr>
<td>1 2 3 6 11 23</td>
<td>...</td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>n</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td></td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td></td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td></td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td></td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td></td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>( n )</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td>( 2n )</td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td></td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td></td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td></td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td></td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>n</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td>2n</td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td>5n − 1</td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td></td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td></td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td></td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>( n )</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td>( 2n )</td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td>( 5n - 1 )</td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td></td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td></td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
### Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>( n )</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td>( 2n )</td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td>( 5n - 1 )</td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td>( 2^n - 1 )</td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td></td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible* and *Mathematics, the Science of Patterns* by Keith Devlin
Abstraction

<table>
<thead>
<tr>
<th>sequence</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 ...</td>
<td>( n )</td>
</tr>
<tr>
<td>2 4 6 8 10 12 ...</td>
<td>( 2n )</td>
</tr>
<tr>
<td>4 9 14 19 24 29 ...</td>
<td>( 5n - 1 )</td>
</tr>
<tr>
<td>1 4 9 16 25 36 ...</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>1 3 7 15 31 63 ...</td>
<td>( 2^n - 1 )</td>
</tr>
<tr>
<td>1 2 3 6 11 23 ...</td>
<td>trees with ( n ) nodes</td>
</tr>
</tbody>
</table>

The On-Line Encyclopedia of Integer Sequences

*The Language of Mathematics: Making the Invisible Visible and Mathematics, the Science of Patterns* by Keith Devlin
Another example — maps

Consider the problem of giving directions from the *Rathaus* to *Karlsplatz* in downtown Munich. An aerial photograph may be handy, but ...
Abstraction

Abstractions are especially important in Computer Science because every program exists largely apart from the physical world.

We now know that electronic technology has no more to contribute to computing than the physical equipment. We now know that a programmable computer is no more and no less than an extremely handy device for realizing any conceivable mechanism without changing a single wire, and that the core challenge for computing science is a conceptual one, viz. what (abstract) mechanisms we can conceive without getting lost in the complexities of our own making.

Complexity

The so-called LAMP stack (Linux OS/Apache internet server/MySQL DBMS/Perl PL) consists of 10 million lines of code, interacting in myriad ways to achieve impressive functionality and performance. This approaches the intellectual complexity of the Saturn V rocket (with three million parts) which took man to the moon. The similarities between these two enormous engineering feats are important: good engineering design practices have been followed; requirements were defined and met; rigorous testing and debugging has taken place. Yet, to date, the LAMP stack is much less well understood than the Saturn V rocket. It is much harder to predict how the LAMP stack will perform under varying conditions and where things might go wrong than it is to consider how the Saturn V may behave in different operating environments.

Morrison and Snodgrass, CACM, 2011
The right abstractions combat complexity, but they take effort.

It is true that we live in a complex world and strive to solve inherently complex problems, which often do require complex mechanisms. However, this should not diminish our desire for elegant solutions, which convince by their clarity and effectiveness. Simple, elegant solutions are more effective, but they are harder to find than complex ones, and they require more time, which we often believe to be unaffordable.

Niklaus Wirth, Turing Award lecture, 1984.
Complexity

Sometimes people make something overly complex on purpose.

*There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies; and the other way is to make it so complicated that there are no obvious deficiencies.*

Charles Anthony Richard Hoare, Turing Award lecture, 1981
Expression versus Visualization

- Abstractions often lead to a model, e.g., list of longitude, latitude pairs representing the shorelines of the continents.
- Scientific visualization of data (models) is an important branch of computer science, e.g., equal-area and conformal projections are different visualizations of the same data.
- Abstractions, models, visualization are all relative to point of view, i.e., what’s important in one context is unimportant to another.
- Visualization is different but related to expression — visualization is “to the mind” and expression “from the mind.” The human mind can perceive graphical information quite readily.
Points on a sphere can be represented by the latitude and longitude.

<table>
<thead>
<tr>
<th>Latitude</th>
<th>Longitude</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.883</td>
<td>-77.033</td>
<td>Washington, D.C.</td>
</tr>
<tr>
<td>55.750</td>
<td>37.700</td>
<td>Moscow, Russia</td>
</tr>
<tr>
<td>-34.667</td>
<td>-58.500</td>
<td>Buenos Aires, Argentina</td>
</tr>
<tr>
<td>19.400</td>
<td>-99.150</td>
<td>Ciudad de Mexico (Mexico City)</td>
</tr>
<tr>
<td>45.417</td>
<td>-75.717</td>
<td>Ottawa, Canada</td>
</tr>
<tr>
<td>-12.050</td>
<td>-77.050</td>
<td>Lima, Per</td>
</tr>
<tr>
<td>-33.450</td>
<td>-70.667</td>
<td>Santiago, Chile</td>
</tr>
<tr>
<td>48.867</td>
<td>2.333</td>
<td>Paris, France</td>
</tr>
<tr>
<td>56.883</td>
<td>24.133</td>
<td>Riga, Latvia</td>
</tr>
</tbody>
</table>
Topological View of Earth (unprojected)

NASA “Blue Marble” image, part of the Visible Earth Catalog
Mercator Projection
Given the longitude east of Greenwich $\lambda$ and the latitude $\phi$ north of the equator, the underlying model of the Mercator projection is given by:

\[
x = R\lambda \\
y = R\ln\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)
\]
Behrmann Cylindrical Equal-Area
Cylindrical Equal-Area

The underlying models of cylindrical equal-area projections:

\[ x = R(\lambda - \lambda_0) \cos \phi_s \]
\[ y = R \sin \phi \sec \phi_s \]

<table>
<thead>
<tr>
<th>$\phi_s$</th>
<th>map projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>Lambert</td>
</tr>
<tr>
<td>30°</td>
<td>Behrman</td>
</tr>
<tr>
<td>37.400°</td>
<td>Trystan Edwards (or 37.383)</td>
</tr>
<tr>
<td>44.138°</td>
<td>Peters (or 45)</td>
</tr>
<tr>
<td>45°</td>
<td>Gall Orthographic</td>
</tr>
<tr>
<td>50°</td>
<td>Balthasart</td>
</tr>
</tbody>
</table>
The underlying model of the Mollweide projection:

\[ x = 2\sqrt{2}R(\lambda - \lambda_0)(\cos \theta)/\pi \]

\[ y = \sqrt{2}R\sin \theta \]

where \( \theta \) is defined for a given \( \phi \) by:

\[ 2\theta + \sin 2\theta = \pi \sin \phi \]
Definitions

**Simple:** readily understood or apprehended (by a person)

**Natural:** obvious (to a person), expected and unaffected

**Complex:** hard to separate, analyze, or solve
**Abstraction:** conceptualization without reference to specific instances. The act of determining the fundamental, essential, important, intrinsic properties disassociated from the specific details.

**Model:** a formal or schematic description of a system, theory, phenomenon

**Visualization:** The act of making a visible presentation of data, particularly a graphical presentation. For example, a graph is a visualization of a function. Virtual reality is another form of visualization.
**Expression:** representing in a medium (as words or programs) or representing in some medium

**To represent:** to portray, exhibit, form an image, to describe as having a specified character or quality, to apprehend by means of an idea

**To denote something:** to find expression in a language of some notion

**Medium:** the means of conveying or transmitting something;

**Language:** System of symbols, signs, sounds, or gestures used for communication

**Notation:** Marks, signs, figures, or characters representing something.
The goal of the field of programming languages is to find natural (to people) expression of computation which is efficiently translatable to computers. What is natural is shaped by psychology, experience, and education. Progress: better notation enables better abstraction, and *vice versa*.
We cut nature up, organize it into concepts, and ascribe significances as we do, largely because we are parties to an agreement to organize it in this way—an agreement that holds throughout our speech community and is codified in patterns of our languages. The agreement is, of course, an implicit and unstated one, but its terms are absolutely obligatory; we cannot talk at all except by subscribing to the organization and classification of data which the agreement decrees.

Die Grenze meiner Sprache bedeuten die Grenzen meiner Welt – Tractatus Logico-Philosophicus 5.6
Die Sprache bestimmt das Denken.
He who controls vocabulary controls thought.

Ludwig Wittgenstein (1889–1951)
The end of Tractatus:
Wovon man nicht sprechen kann, darüber muss man schweigen
We do not learn language by reading a dictionary, and we do not think or speak in terms of dictionary definitions. Meaning is always more fluid. Nevertheless, we are hemmed in, even trapped, by common usage. Senses we wish to evade entrap us. The greatest escape route is not only humor, but poetry, or art in general. Art does not, of course, liberate us completely from meaning, but it gives a certain measure of freedom, provides elbow room.

Charles Rosen, the pianist who died in 2012, in an essay entitled “Freedom and Art.”
In making our adjustments to our particular programming language, we can easily become attached to it simply because we now have to much invested in it. Most people would prefer almost any amount of pain to giving up the familiarity of some constant companion for an unknown quantity. We see this effect when we try to teach a programmer his second language. Teaching the first is no great problem, for he has no investment in any other. By the time he has learned two or more, he is aware that more things exist in this world than he has dreamed of. But letting go of the first is, to him, just a promise of pain with no promise of compensating pleasure.

The Psychology of Computer Programming by Gerald M. Weinberg
Examples of the power of a well-contrived notation to condense into small space a meaning which would—in ordinary language—require several lines, or even pages, can hardly have escaped the notice of most of my readers: ... instead of creating any obscurity, the expressions are far more readily understood than if they were written at length.

Charles Babbage, 1827.

- Multiplication in Roman numerals
- FORTRAN, rocket launch
- C break statement and the AT&T telephone outage
Value of Notation

By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race. Before the introduction of the Arabic notation, multiplication was difficult, and the division even of integers called into play the highest mathematical faculties. Probably nothing in the modern world would have more astonished a Greek mathematician than to learn that . . . a large proportion of the population of Western Europe could perform the operation of division for the largest numbers. This fact would have seemed to him a sheer impossibility. . . .

Our modern power of easy reckoning with decimal fractions is the almost miraculous result of the gradual discovery of a perfect notation.

purposes, however, the history of the notation is a detail. The interesting point to notice is the admirable illustration which this numeral system affords of the enormous importance of a good notation. By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race. Before the introduction of the Arabic notation, multiplication was difficult, and the division even of integers called into play the highest mathematical faculties. Probably nothing in the modern world would have more astonished a Greek mathematician than to learn that, under the influence of compulsory education, the whole population of Western Europe, from the highest to the lowest, could perform the operation of division for the largest numbers. This fact would have seemed to him a sheer impossibility. The consequential extension of the notation to decimal fractions was not accomplished till the seventeenth century. Our modern power of easy reckoning with decimal fractions is the almost miraculous result of the gradual discovery of a perfect notation.

Mathematics is often considered a difficult and mysterious science, because of the numerous symbols which it employs. Of course, nothing is more incomprehensible than
Enough analogy with linguistics and geography, what are the important abstractions in computation?
Why do we care? The abstractions in computing that give rise to useful mediums of expression are clearly useful for designing programming languages. Sometimes (most of the time) we see these abstractions clearly only in retrospect.
One way of organizing these abstractions gives rise to three kinds of programming languages. Often we call these programming language paradigms.
Computational Paradigms

- **Imperative:** :=, goto, if
- **Functional:** beta reduction

\[(\lambda x. b) a \rightarrow b[x := a]\]

- **Logic:** resolution principle

\[Q_1, \ldots, Q_n \leftarrow P_1, P_2, \ldots, P_k \quad R_1, R_2, \ldots, R_m \leftarrow S_1, \ldots, S_l\]

\[\begin{array}{c}
(Q_1, \ldots, Q_n, R_2, \ldots, R_m)\sigma \leftarrow (P_2, \ldots, P_k, S_1, \ldots, S_n)\sigma
\end{array}\]
Imperative languages are characterized as having an implicit state that is modified (i.e., side affected) by constructs (i.e., commands) in the source language. As a result, such languages generally have a notion of sequencing (of the commands) to permit precise and deterministic control over the state. Most, including the most popular, languages in existence today are imperative.

Other Aspects of Programming

Computation is the most general purpose a programming language has, but other aspects are important, too.

- Declarative programming – what versus how
- Object-oriented programming – organization of data
- Structured programming – style, discipline
- Distributed computing – external interaction
- Parallel programming – implementation
- Database programming – persistence of data
Kinds of Programming Languages

There are many ways to organize the different kinds of programming languages.

- Imperative, functional, logic
- Object-oriented programming languages
- Database programming languages
- Scripting programming languages
  Scott: “most actual uses of computers require the coordination of multiple programs.” Cf. Scott, 2ed, Chapter 13: Scripting Languages.
  ALSU: “interpreted languages with high-level operators designed for ‘gluing together’ computations.”
  Ousterhout: “Scripting languages are designed for ‘gluing’ applications; they use typeless approaches to achieve a higher level of programming and more rapid application development.”
- Esoteric programming languages (“esolang”), eg., INTERCAL, Whitespace, Thue
Compare with Sebesta Section 1.5 Language Categories.

page 23. 1.5 Language Categories: “four bins: imperative, functional, logic, and object oriented.”