16. Programming Languages

The Tower of Babel

A story about the origins of multiple languages

- [After the flood] "The whole earth was of one language and one speech."
- They built a city and tower at Babel, believing that with a single language, people will be able to do anything they imagine.
- Yahweh disagrees and "confounds the language of all the earth"
- Why?

Proliferation of cultural differences (and multiple languages) is one basis of civilization.

Several ways to solve a transportation problem

- Popular languages
  - Java in context
  - Object-oriented programming
  - Type checking
  - Functional programming
Several ways to solve a programming problem

You can write Java code.

```java
public class ThreeSum {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int[] a = new int[N];
        for (int i = 0; i < N; i++)
            a[i] = StdIn.readInt();
        for (int i = 0; i < N - 1; i++)
            for (int j = i + 1; j < N; j++)
                if (a[i] + a[j] + a[k] == 0) {
                    StdOut.println(a[i] + a[j] + a[k]);
                }
    }
}
```

You can also write C code.

```c
#include <stdio.h>
#include <stdlib.h>
main(int argc, char argv[])
{
    int N = atoi(argv[1]);
    int *a = malloc(N*sizeof(int));
    int i, j, k;
    for (i = 0; i < N; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < N - 1; i++)
        for (j = i + 1; j < N; j++)
            if (a[i] + a[j] + a[k] == 0)
                printf("%d %d %d\n", a[i], a[j], a[k]);
}
```

A big difference between C and Java (there are many!)

- **NO DATA ABSTRACTION**
  - No objects in C.
  - A C program is a sequence of static methods.

- **C++ (Stroustrup 1989)**
  - Adds data abstraction to C.
  - "C with classes".
  - Embodies many OOP innovations.

"There are only two kinds of programming languages: those people always [gripe] about and those nobody uses."

− Bjarne Stroustrup
# C++

You can also write C++ code.

### Example 1. Use C++ like C.

**Notable differences**
- library conventions
- standard input idiom
- standard output idiom
- pointer manipulation (stay tuned)

```cpp
#include <iostream>
#include <cstdlib>
int main(int argc, char *argv[]) {
    int N = atoi(argv[1]);
    int *a = new int[N];
    int i, j, k;
    for (i = 0; i < N; i++)
        std::cin >> a[i];
    for (j = i+1; j < N; j++)
        for (k = j+1; k < N; k++)
            if (a[i] + a[j] + a[k] == 0)
                std::cout << a[i] << " " << a[j] << " " << a[k] << std::endl;
    delete[] a;
    return 0;
}
```

### Example 2. Use C++ like Java to implement the symbol table ADT.

**Challenges**
- libraries/idioms
- pointer manipulation
- templates (generics)

```cpp
template <class Item, class Key>
class ST {
    struct node {
        item *item;
        node *left, *right;
    };
    typedef node link;
    link *head;
    item searchRN(link x, Key key) {
        if (!x) return 0;
        if (key == x->item.key) return x;
        if (key < x->item.key) return searchRN(x->left, key);
        else return searchRN(x->right, key);
    }
};
```

---

# Python

You can also use Python like a calculator.

### Example 1. Use Python like a calculator.

```python
>>> 2+2
4
>>> (1 + sqrt(5))/2
```

---

A big difference between C/C++ and Java (there are many!)

**C/C++: YOU are responsible for memory allocation**
- Programs manipulate pointers.
- System provides memory allocation library.
- Programs explicitly call methods that "allocate" and "free" memory for objects.
- Pitfall: "memory leaks".

**Java: Automatic "garbage collection"**
- System keeps track of references.
- System manages memory use.
- System reclaim memory that is no longer accessible from your program.

**Fundamental challenge. C/C++ code that manipulates pointers is inherently unsafe.**
Python

You can also write Python code.

**Example 2.** Use Python like Java.

**Notable differences**
- No braces (indents instead).
- No type declarations.
- Array creation idioms.
- I/O idioms.
- for (iterable) idioms.

Matlab

You can write Matlab code.

**Example 1.** Use Matlab like Java.

**Example 2 (more typical).** Use Matlab for matrix processing.

\[
\begin{bmatrix}
1 & 3 & 5 \\
2 & 4 & 7
\end{bmatrix}
\begin{bmatrix}
5 & 8 \\
3 & 9 \\
4 & 0
\end{bmatrix} =
\begin{bmatrix}
24 & 35 \\
30 & 52
\end{bmatrix}
\]
Big differences between Matlab and C/C++/Java/Python (there are many!)

1. **MATLAB IS NOT FREE.**
2. Most Matlab programmers use only ONE data type (matrix).

**Example.** Matlab code \( i = 0 \) means
"redefine the value of the complex number \( i \) to be a 1-by-1 matrix whose entry is 0"

**Notes**
- Matlab is written in Java.
- The Java compiler and interpreters are written in C.
  [Modern C compilers are written in C.]
- Matrix libraries (written in C) are accessible from C/C++/Java/Python.

**Reasonable approaches**
- Use Matlab as a "matrix calculator" and data analysis (if you own it).
- Convert to or use Java/C/C++ if you want to do anything else.

---

16. Programming Languages

- **Popular languages**
- Java in context
- Object-oriented programming
- Type checking
- Functional programming

**Why Java?** [revisited from second lecture]

**Java features**
- Widely used.
- Widely available.
- Continuously under development since early 1990s.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.

**Java economy**
- Mars rovers.
- Cell phones.
- Blu-ray Disc.
- Web servers.
- Medical devices.
- Supercomputing.
- …
Why do we use Java in this course?

<table>
<thead>
<tr>
<th>language</th>
<th>widely used</th>
<th>widely available</th>
<th>full set of modern abstractions</th>
<th>modern libraries and systems</th>
<th>automatic checks for bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>C++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>maybe</td>
<td>✓</td>
</tr>
<tr>
<td>Java</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MATLAB</td>
<td>✓</td>
<td>$</td>
<td>maybe*</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>maybe</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

* OOP recently added but not embraced by most users

Why learn another programming language?

Good reasons to learn a programming language

- Offers something new.
- Need to interface with co-workers.
- Better than Java for the application at hand.
- Provides an intellectual challenge
- Opportunity to learn something about computation.
- Introduces a new programming style.

Something new: a few examples

1960s: Assembly language
- symbolic names
- relocatable code

1970s: C
- “high-level” language
- statements, conditionals, loops
- machine-independent code
- functions and libraries

1990s: C++/Java
- data abstraction (OOP)
- extensive libraries

2000s: AJAX/PHP/Ruby/Flash
- scripting
- libraries for web development

Programming styles

<table>
<thead>
<tr>
<th>style</th>
<th>execution model</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>procedural</td>
<td>step-by-step instruction execution usually compiled</td>
<td>C++</td>
</tr>
<tr>
<td>scripted</td>
<td>step-by-step command execution usually interpreted</td>
<td>python</td>
</tr>
<tr>
<td>special-purpose</td>
<td>optimized around certain data types</td>
<td>MATLAB</td>
</tr>
<tr>
<td>object-oriented</td>
<td>focus on objects that do things</td>
<td>Java</td>
</tr>
<tr>
<td>functional</td>
<td>executable functions can be arguments or return values, or stored as data</td>
<td>OCaml, Haskell</td>
</tr>
</tbody>
</table>
16. Programming Languages

- Popular languages
- Java in context
- Object-oriented programming
- Type checking
- Functional programming

Object-oriented programming

A different philosophy
- Software is a simulation of the real world.
- We know (approximately) how the real world works.
- Design software to (approximately) model the real world.

Procedural programming
- Tell the computer to do this.
- Tell the computer to do that.

Objected oriented programming (OOP)
- Programming paradigm based on data types.
- Identify things that are part of the problem domain or solution.
- Things in the world know something: instance variables.
- Things in the world do something: methods.

Why OOP?

Essential questions
- Is my program easy to write?
- Is it easy to find errors and maintain my program?
- Is it correct and efficient?

Essential features of OOP
- Encapsulation to hide information to make programs robust.
- Type checking to avoid and find errors in programs.
- Libraries to reuse code.
- Immutability to guarantee stability of program data.

Does OOP make it easy to write and maintain correct and efficient programs?
- Difficult for you to know, because you haven’t programmed in another style.
- Ongoing debate among experts intensifies as time goes on.
- Meanwhile, millions of people (including YOU) are reaping the benefits of OOP.

Warning: OOP involves deep, difficult, and controversial issues. Further study may be fruitful, but is likely to raise more questions than answers!
OOP pioneers

Kristen Nygaard and O.J. Dahl. (U. Oslo 1960s)
- Invented OOP for simulation.
- Developed Simula programming language.
- Studied formal reasoning about OOP programs.

Alan Kay. (Xerox PARC 1970s)
- Developed Smalltalk programming language.
- Promoted OOP for widespread use.
- Computer science visionary.

Barbara Liskov. (MIT 1970s)
- Developed CLU programming language.
- Pioneered focus on data abstraction.
- Research provided basis for Java, C++, ...

Alan Kay: a computer science visionary

1970s
- Typical "mainframe" computer: IBM 360/50
- First PC: Xerox Alto
- Alan Kay's vision for the future
- Dynabook prototype
- Key feature: OOP software (Smalltalk)

2010s
- Modern personal computer
- MacBook Air
- Key feature: OOP software (Objective C)

"The best way to predict the future is to invent it." (1971)
"The computer revolution hasn't happened yet." (1997)

Still relevant today!

— Alan Kay

16. Programming Languages

- Popular languages
- Java in context
- Object-oriented programming
- Type checking
- Functional programming

http://introcs.cs.princeton.edu
**Type checking**

Static (compile-time) type checking (e.g. Java)
- All variables have declared types.
- System checks for type errors at compile time.

Dynamic (run-time) type checking (e.g. Python)
- Values, not variables, have defined types.
- System checks for type errors at run time.

Q. Which is best?

A. Religious wars ongoing!
- Static typing worth the trouble?
- Compiled code more efficient?
- Type-checked code more reliable?
- Advanced features (e.g. generics) too difficult to use with static typing?

---

**Example: Diametrically opposed points of view**

**Issue.** Type checking or automated program testing?

"Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence."
— Edgar Dijkstra (1969)

"Since static type checking can't cover all possibilities, you will need automated testing. Once you have automated testing, static type checking is redundant."
— Python blogger (2009)

---

**A letter from Dave Walker**

Dear random python blogger:

Why don’t you think of static type checking as a complementary form of completely automated testing to augment your other testing techniques? I actually don’t know of any other testing infrastructure that is as automated, fast and responsive as a type checker, but I’d be happy to learn.

By the way, type checking is a special kind of testing that scales perfectly to software of arbitrary size because it checks that the composition of 2 modules is ok based only on their interfaces, without re-examining their implementations. Conventional testing does not scale the same way. Also, did you know that type checking is capable of guaranteeing the absence of certain classes of bugs? That is particularly important if you want your system to be secure. Python can’t do that.

dpw (in mail to rs)

---

**Programming folklore: Hungarian type system**

Early programming languages had little support for types.

Hungarian type system (Charles Simonyi, 1970s)
- Encode type in first few characters of variable name.
- 8 character limit? Leave out the vowels, truncate.

Example. arru8Fbn

- array of 8-bit integers (unsigned)
- variable name short for Fibonacci

An advantage: Can "type check" while reading code.
A disadvantage: shrt vwl-lss vrbl nms.

Used in first version of Microsoft Word (and extensively before that time).
Lesson. Type-checking has always been important in large software systems.

Charles Simonyi
Introduced OOP to Microsoft
16. Programming Languages

- Popular languages
- Java in context
- Object-oriented programming
- Type checking
- Functional programming

Functional programming

Q. Why can’t we use functions as arguments in Java programs?

A. Good question. We can, but doing so requires interfaces and is cumbersome.

Functional programming is a function-oriented programming style.
- Functions are first-class entities (can be arguments and return values of other functions or stored as data).
- On-demand execution model.
- “What” rather than “how”.

Advantages of functional programming
- Often leads to more compact code than alternatives.
- More easily admits type system that can result in “provably correct” code.
- More easily supports concurrency (programming on multiple processors).

Disadvantage. Can be more difficult to focus on performance.
Functional programming example

A Python program that prints a table of squares.

def square(x):
    return x**2

def table(f, sequence):
    for x in sequence:
        print x,
        print f(x)

table(square, range(10))

Functions that operate on functions

Functions as first-class objects admit compact code for powerful operations.

Example 1. The MAP operation takes a function and a list as arguments. MAP(f, sequence) is the result of replacing every x in sequence by f(x).

def square(x):
    return x**2

def odd(x):
    return 2*x + 1

print map (odd, range(10))
print map (square, range(10))

Functions that operate on functions

Functions as first-class objects admit compact code for powerful operations.

Example 2. The REDUCE operation takes a function and a list as arguments. REDUCE(f, L) is \text{f(car}(L)), REDUCE(f, \text{cdr}(L)).

def plus(x, y):
    return x + y

def odd(x):
    return 2*x + 1

print reduce(plus, map(odd, range(10)))

Why learn functional programming?

Good reasons to learn a programming language

• Offers something new.
• Need to interface with co-workers.
• Better than Java for the application at hand.
• Provides an intellectual challenge.
• Opportunity to learn something about computation.
• Introduces a new programming style.

Modern applications

• Communications systems
• Financial systems
• Google MapReduce

Deep and direct connections to theoretical CS (stay tuned).

Warning. Functional programming may be addictive.

Intro CS at MIT was taught in Scheme (a functional language) for decades

Functional Programming jobs 74747474747474
The Tower of Babel

A story about the origins of multiple languages

- After the flood
  - "The whole earth was of one language and one speech."
- They built a city and tower at Babel, believing that with a single language, people will be able to do anything they imagine.
- Yahweh disagrees and "confounds the language of all the earth"
- Why?

Image from cover of Programming Languages by Joan Sommerville (1989). Already 120+ languages.

An apt metaphor:

- Would a single programming language enable us to do anything that we imagine?
- Is the proliferation of languages a basis of civilization in programming?