Programming Language Families

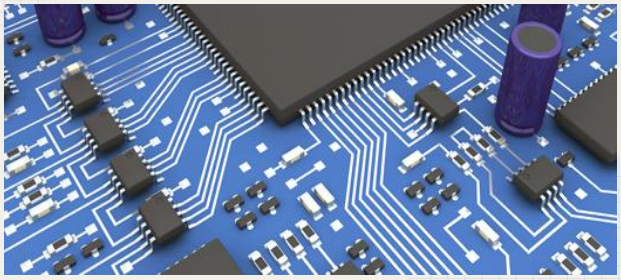
Presentation Overview

Introduction

- Declarative vs Imperative
  - What vs How
  - Multiparadigm
  - Advantages/Disadvantages in general
- Four language families (general overview)
  - Procedural
  - Functional
  - Object Oriented
  - Logical

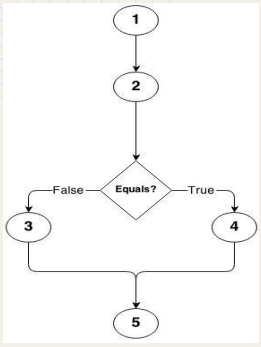
Presentation Overview

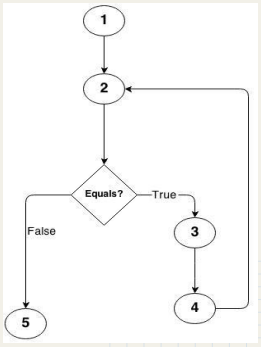
Procedural Paradigm



Procedural Programming

- Procedural programming = Imperative programming with subroutines
- Step by step, close to low-level programming
- Based on procedure (method) calls
  - Pass value, return value
- Simple programming
  - No encapsulation of functionality and data

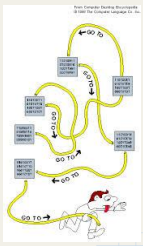




Procedural: Advantages

- Fast compilation

- Debuggable



- Easy to read and understand
- Portable source code

Procedural Example

```
int Sum(int a, int b)
{
        return a+b;
}

int main()
{
        int sum = Sum(5,7);
        return 0;
}
```

```
class Summation()
{
        private sum;

        public static void main (String [] args)
        {
                int sum=Sum(5,7);
        }

        public int Sum(int a, int b)
        {       return a+b;     }
}
```


Procedural Void example

```
void printToScreen(char* text)
    {
     printf("Message is %s", text);
    }

    int main()
    {
     printToScreen("Procedural programming paradigm");
     return 0;
    }
```


Procedural Example

```
int sum=0;

for (int i=0; i<150; i++)
        if (i % 5!=0)
                sum=sum+i;
```


```
int sum = Enumerable.Range(0,150).Where(i => i%5 != 0).Sum();
```


Procedural: Disadvantages

- Difficult implementation of fuzzy conditions
- Extremely difficult and inefficient to solve large scale problems

- No data security
- Difficult to maintain if code growth larger

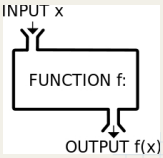
Presentation Overview

- Introduction
- Procedural Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Functional Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Object Oriented Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Logical Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Conclusion
- Questions

Functional Languages

Functional: definition

- Functions are first class objects
- Ability of defining and manipulating the functions from within other functions
- Stateless
- No looping, recursion



Functional: types

Two types of functional programming languages:

- Pure
- Impure

Functional: Advantages

```
qsort :: [a] --> [a]
 qsort [] = []
 qsort (h:t) = qsort [y | y <-- t, y < h ] ++ [h] ++ qsort [y | y <-- t, y >= h]
```


Functional: Advantages

```
void quickSort(int arr[], int left, int
   right)
    {
     int i = left, j = right;
     int tmp;
     int pivot = arr[(left + right) / 2];

     /* partition */
     while (i <= j) {
          while (arr[i] < pivot)
               i++;
while (arr[j] > pivot)
          j--;
 if (i <= j) {
          tmp = arr[i];
          arr[i] = arr[j];
          arr[j] = tmp;
          i++;
          j--;
     }
}
/* recursion */
if (left < j)
    quickSort(arr, left, j);
if (i < right)
       quickSort(arr, i, right);
}
```

Functional: Advantages

- Lazy evaluation

- No side effects

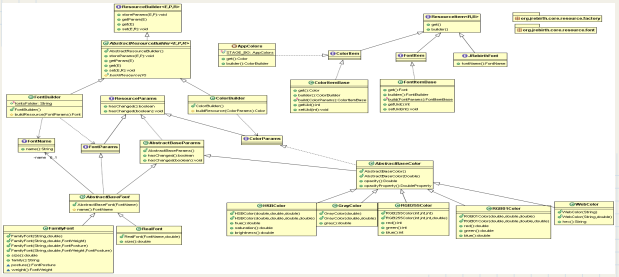
Functional: Disadvantages

- Not a lot of support for new learners. (compared to other languages)
- Code is not very much self-documented (hard to read)
- Statistically, programmers usually take longer to learn this language.
- Strong mathematical background is usually needed to produce and/or understand functional code.
- Hard to use functional programming for problems that have multiple input/ output.

Presentation Overview

- Introduction
- Procedural Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Functional Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Object Oriented Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Logical Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Conclusion
- Questions

Object-Oriented Programming



Object-Oriented Programming

- It's more of a paradigm than a language type
  - i.e. C could be use as object-oriented (structs with function pointers)
- Imitates real world
- Very Popular
- Goals:
  - Increased understanding
  - Ease of maintenance
  - Ease of evolution

OO: Characteristics

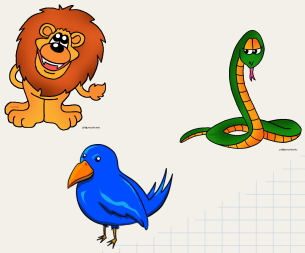
- Classes and Objects
- Abstraction
- Encapsulation
- Modularity
- Hierarchy
  - Inheritance
  - Aggregation
- Many more...

OO: Classes and Objects

- Class: an object type

ANIMAL

- Object: an instance of a class

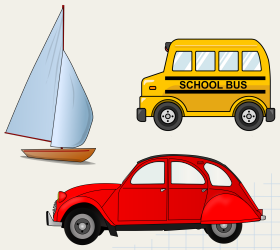


OO: Classes and Objects

- Class: an object type

VEHICLE

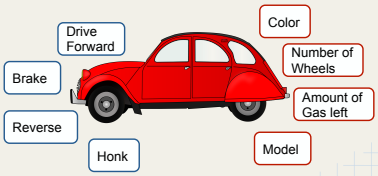
- Object: an instance of a class



OO: Objects

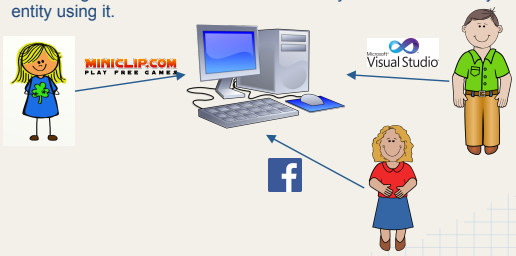
- Contain:
  - Functionality (Functions)
  - Characteristics (Data)

```
class Vehicle{
    public int color;
    public int numWheels;
    private double gasLeft;
    public string model;

    public void driveForward();
    public void reverse();
    public void brake();
    public void honk();

    public double get_gasAmount();

    private double calculateGas();
    //...
```



OO: Abstraction

Deliberate ignorance of the **whole** functionality/character of an object by the entity using it.



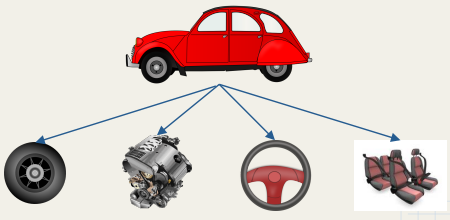
OO: Encapsulation

Object encapsulates characteristics and intricacies of its functionality. (provides a simple interface for other objects)

This girl did not take CPSC 441, but can still use the browser to connect to the internet.

OO: Modularity

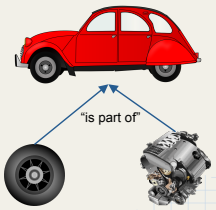
Separate complex modules into smaller parts.



OO: Hierarchy

A way to rank (order) abstractions
- Inheritance:

- Aggregation:

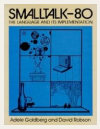


OO: Advantages

- Very close to real world interaction.
  - Best when problem resembles real world (has clearly defined entities)
- If **understandable**, it is **easily maintainable** and **easily changeable.**
- Huge popularity = huge support online.
- Design Patterns

Examples

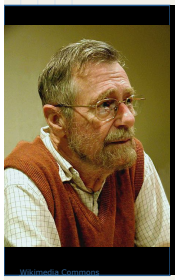
- Pure OO (even chars and integers are treated as objects):



Ruby

- OO with some procedural:

Why OO sucks (1)



(For those who have wondered: I don't think object-oriented programming is a structuring paradigm that meets my standards of elegance.)"

-Edsgar Dijkstra

From the Keynote address to be given on 1 March 1999 at the ACM Symposium on Applied Computing at San Antonio, TX
https://www.cs.utexas.edu/users/EWD/transcriptions/EWD12xx/EWD1284.html

Why OO sucks (2)

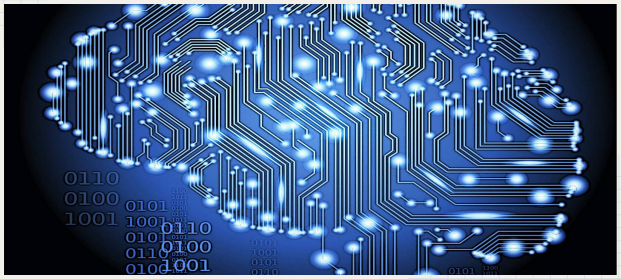
Lacks Normalization Rules
- OO has nothing equivalent to relational Normal Form rules and guidelines to produce consistency and reduce known duplication

Grouping Functions and Data Structures together
- Functions take, do something to, then return data structures
- Data structures just … are
- They are fundamentally different so why are they 'encapsulated' together
- reduces cross-language sharing of protocols and data
- Fails Divide and Conquer of data and operations

And many, many more criticisms… for a later time perhaps

Presentation Overview

Logical Programming



Logic Programming

- Dijkstra mentioned predicate calculus being a simple and elegant device. Logic programming is based on formal logic, making use of predicate calculus
- A logic program has a 'knowledge base': set of 'facts' that are assumed to be true, and 'inferences' which are conditionally true (ie. A --> B)
- Inference is the derivation of logical conclusions from premises assumed to be true

Logic: Prolog Example

- Prolog is the most popular Logic PL. The program is a collection of facts, and is used interactively.

Prolog Code:

```
female(alice).
female(victoria).

male(albert).
male(edward).

parents(edward, victoria, albert).
parents(alice, victoria, albert).

sister_of(X,Y) :- female(X), parents(X,M,F),
                  parents(Y,M,F), X\=Y

organism :- male(X).
organism :- female(X).
```

Prolog Queries:

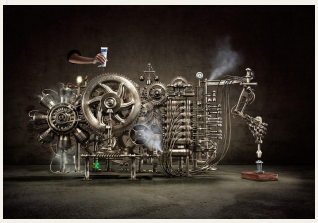
```
?- female(alice).
true.

?- male(X).
X = albert ;
X = edward.

?- sister_of(X,Y).
X = alice,
Y = edward .

?- organism(X).
X = alice ;
X = victoria ;
X = albert ;
X = edward.
```


Logic: So what is it good for?

- Is inherently the premiere choice for Artificial Intelligence
  - confirm or deny queries using a knowledge base
  - derive *logical* conclusions from a provided knowledge base
- Other applications:
  - complex scheduling problems
  - warehousing problems
  - database problems

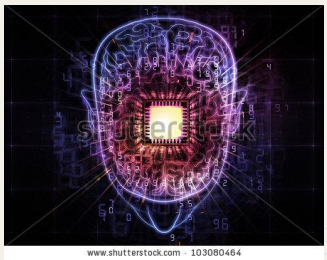
Logic = boo

- Complex compilers compared to procedural compilers



Logic = boo

- Requires different method of thinking



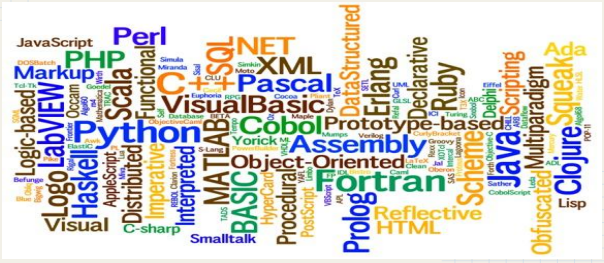
Logic = boo

- Decent understanding of recursion
- Limited uses in industry, and applications are comparatively small
- still a “research baby”

Presentation Overview

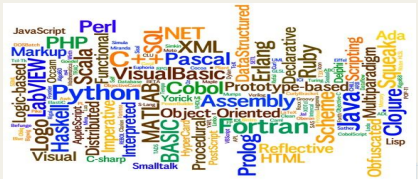
- Introduction
- Procedural Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Functional Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Object Oriented Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Logical Languages
  - Introduction
  - Advantages
  - Examples
  - Disadvantages
- Conclusion
- Questions

Conclusion

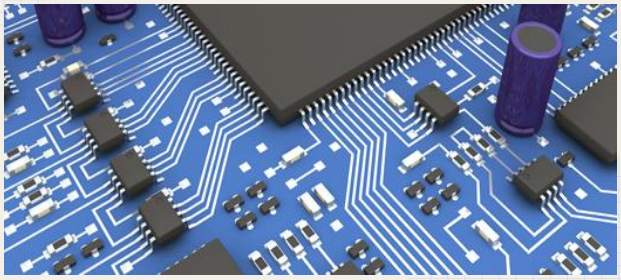


Conclusion

- One can do formal proofs for functional programming languages but not in procedural paradigm
- A simple problem is easily implemented in procedural paradigm rather than in object-oriented paradigm

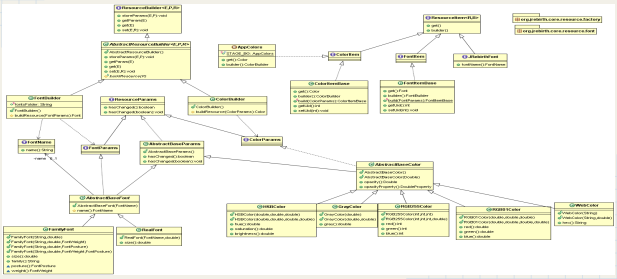


Procedural Programming

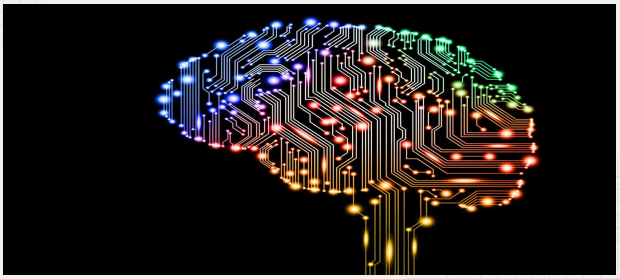


Functional Programming

Object-Oriented Programming



Logical Programming



Conclusion

- Functional and Logical languages are very important in research
  - Example: Haskell - functional language was developed as research language
- Object-oriented and Procedural languages are very popular in industry
  - Example: Python, Java, C, C++ are very commonly used programming languages nowadays

Need the Right Tool

Questions?