FIT ICT Software Development

Lecture 1

Programming Paradigms – history and overview

Lecturer : Charles Tyner

Early Programming - 1

1822 – Charles Babbage develops his Difference Engine, an analogue computer. This machine could only be made to execute tasks by changing the gears which executed the calculations.

1840 – Babbage gives a seminar on his (theoretical) Analytical Engine. Written up in French by an Italian engineer (and future Italian PM) Luigi Menabrea.

1842-3 – Ada, Lady Lovelace, an assistant of Babbage, is commissioned to translate back into English. She augments the paper with notes, which are added to the translation.

1953 - Lovelace's notes on Babbage's Analytical Engine are republished. The engine has now been recognised as an early model for a computer and her notes as a description of a computer and software.

Early Programming - 2

Lovelace's notes were labelled alphabetically from A to G.

In note G, she describes an algorithm for the Analytical Engine to compute Bernoulli numbers. It is considered the first algorithm ever specifically tailored for implementation on a computer, and Ada has often been cited as the first computer programmer for this reason.

The engine was never completed so her code was never tested.

Early Programming - 3

World War 2 - Allies

1943 – US Army commissions ENIAC, the first electronic general-purpose computer. However, could only be "programmed" by presetting switches and rewiring the entire system for each new calculation.

1943 – Tommy Flowers, Post Office Engineer seconded to Bletchley Park, builds Colossus to help with cryptanalysis of Nazi codes. Colossus was the first of the electronic digital machines with (very limited) programmability.

- It had no internally stored programs. To set it up for a new task, the operator had to set up plugs and switches to alter the wiring.
- Colossus was not a general-purpose machine, being designed for a specific cryptanalytic task involving counting and Boolean operations.

Early Programming - 4

World War 2 - Axis

The entire effort resided in one person – Konrad Zuse

1936-38 Built the Z1, which contained some 30,000 metal parts and never worked well due to insufficient mechanical precision.

1939 – Drafted into military service and given resources to build Z2, an improved version of Z1 that used telephone relays.

1941 - Zuse presents Z3, a binary 22-bit floating point calculator featuring programmability with loops but without conditional jumps, with memory and a calculation unit based on telephone relays.

1944 – Work on Z4 abandoned by Nazis as "strategically unimportant", and not resumed until 1949. In 1950, one was ordered by the Federal Institute of Technology, Zurich, making it the first commercial computer in the world.

Early Programming - 5

World War 2 - Axis

While working on his Z4 computer, Zuse realised that programming in machine code was too complicated.

He started working on a PhD thesis containing ground-breaking research years ahead of its time, mainly the first high-level programming language, Plankalkül ("Plan Calculus") and, as an elaborate example program, the first real computer chess engine.

Thanks to his various machines and his development of a programming language, his Nazi past notwithstanding, Zuse has often been regarded as the inventor of the modern computer. In truth, though, his contribution was not appreciated until many years later, and in the interim, others made similar breakthroughs.

Early Programming - 6

1945 - John Von Neumann working at the Institute for Advanced Study developed two important concepts that directly affected the path of computer programming languages:

The first concept became known as "shared-program technique".

This technique states that the actual computer hardware should be simple and not need to be hand-wired for each program.

Instead, complex instructions should be used to control the simple hardware, allowing it to be reprogrammed much faster.

Early Programming - 7

The second concept was also extremely important to the development of programming languages. Von Neumann called it "conditional control transfer".

This idea gave rise to the notion of subroutines, or small blocks of code that could be jumped to in any order, instead of a single set of chronologically ordered steps for the computer to take.

The second part of the idea stated that computer code should be able to branch based on logical statements such as IF (expression) THEN, and looped such as with a FOR statement.

"Conditional control transfer" gave rise to the idea of "libraries," which are blocks of code that can be reused over and over.

Early Programming - 8

1949 - the language Short Code appears. It is the first computer language for electronic devices.

It requires the programmer to change its statements into 0's and 1's by hand.

1951 - Grace Hopper writes the first compiler, A-0. A compiler is a program that turns the language's statements into 0's and 1's for the computer to understand. This leads to faster programming, as the programmer no longer has to do the work by hand.

1957 – FORTRAN (**FOR**mula **TRAN**slating system)

1958 – ALGOL (**ALGO**rithmic **L**anguage) – ancestor of Pascal, C, C++ and Java

1959 – COBOL (**CO**mmon **B**usiness-**O**riented **L**anguage)

Early Programming - 9

1968 – Pascal, designed specifically as a teaching tool but contains the innovative CASE statement and makes extensive use of pointers.

1972 – C developed by Dennis Ritchie of Bell Labs, New Jersey.
Implements all the innovations of Pascal except its readability.
Built to be fast and powerful at the expense of being hard to read.

1983 – Bjarne Stroustroup , influenced by introduction and growing popularity of concept of Object Oriented Programming, develops extensions to C known as “C With Classes”, released as C++

1995 – Sun Microsystems release Java, a truly object-oriented language that implements advanced techniques such as true portability of code and garbage collection.

A Look At Some Major Paradigms

- Many paradigms
- A lot of overlap between them
- Programs and programming languages can have features of multiple paradigms
- Following slides consider a subset. For more information, see Wikipedia article "Comparison of Programming Paradigms"

Imperative Programming

- Control flow in imperative programming is *explicit*: commands show *how* the computation takes place, step by step.
- Each step affects the global **state** of the computation.
- Main characteristics include
  - direct assignments
  - common data structures
  - global variables
  - Extensive use of goto statements

Imperative Programming

- Earliest imperative languages were the machine languages of the original computers.
- FORTRAN developed by John Backus at IBM starting in 1954
- first major programming language to remove the obstacles presented by machine code in the creation of complex programs.
- FORTRAN was a compiled language that allowed named variables, complex expressions, subprograms, and many other features now common in imperative languages.
- late 1950s and 1960s, ALGOL was developed to allow mathematical algorithms to be more easily expressed

Imperative Programming

```
    result = []
    i = 0
start:
    numPeople = length(people)
    if i >= numPeople goto end
    p = people[i]
    nameLength = length(p.name)
    if nameLength <= 5 goto next
    upperName = toUpper(p.name)
    addToList(result, upperName)
next:
    i = i + 1
    goto start
end:
    return sort(result)
```

Structured Programming

- Structured programming is a kind of imperative programming where the control flow is defined by nested loops, conditionals, and subroutines, rather than via gotos
- Variables are generally local to blocks (have lexical scope)
- Main characteristics include
  - Structograms - graphical representations of programs
  - indentation
  - no, or limited, use of goto statements

Structured Programming

- Early languages emphasizing structured programming: Algol 60, PL/I, Algol 68, Pascal, C, Ada 83, Modula, Modula-2.
- Structured programming as a discipline is sometimes though to have been started by a famous letter by Edsger Dijkstra entitled Go to Statement Considered Harmful.

Structured Programming

```
result = [];
for i = 0; i < length(people); i++ {
    p = people[i];
    if length(p.name)) > 5 {
        addToList(result, toUpper(p.name));
    }
}
return sort(result);
```

Procedural Programming

- Derived from structured programming
- Procedures, also known as methods, functions, routines or sub-routines, contain a series of computational steps to be carried out.
- Procedures can be carried out during any point of the program, sometimes other procedures can call another procedure during it's cycle of running

Procedural Programming

- Main characteristics include
  - local variables
  - sequence
  - selection
  - iteration
  - modularization
- Programming libraries of pre-built codes , routines, sub-routines and functions which can be used at any time by the program and its users are provided
- Languages include C, C++, Lisp, PHP, Python

Procedural Programming

- Code would look similar to that of Structural paradigm.
- Overall look of program looks very different, as procedures help to modularise.

Object Oriented Programming

- Based on the sending of messages to objects.
- Objects respond to messages by performing operations.
- Messages can have arguments, so "sending messages" looks a lot like calling subroutines.
- A society of objects, each with their own "local memory" and own set of operations has a different feel than the "monolithic processor and single shared memory" feel of non object oriented languages.

Object Oriented Programming

- first object oriented language was Simula-67
- Smalltalk followed soon after as the first "pure" object-oriented language.
- Many languages designed from the 1980s to the present have been object-oriented, notably C++, Modula-3, Ada 95, Java, C#, Ruby.

Object Oriented Programming

```
result = []
for p in people {
    if p.name.length > 5 {
        result.add(p.name.toUpper);
    }
}
return result.sort;
```

Event-Driven Programming

- Flow of program execution is determined by *events*
- Designed to detect events as they occur, and then deal with them using an appropriate *event-handling procedure*
- Event-driven programs can be written in any programming language, though some languages provide better tools
- Virtually all object-oriented and visual languages support event-driven programming.

Summary

Paradigm	Description	Main characteristics	Examples
Imperative	Computation as statements that directly c hange a program state (datafields)	Direct assignments, common data structures, global variables	C, C++, Java, PHP, Python
Structured	A style of imperative programming with more logical program structure	Structograms, indentation, either no, or limited use of, gotostatements	C, C++, Java, Python
Procedural	Derived from structured programming, based on the concept of modular programming or the procedure call	Local variables, sequence, selection, iteration, and modularization	C, C++, Lisp, PHP, Python
Object-oriented	Treats datafields as objects m anipulated through pre-defined methods only	Objects, methods, message passing, information hiding, data abstraction	Lisp, C++, C#, Eiffel, Java, PHP, Python, Ruby, Scala
Event-driven	Program flow is determined mainly by events, such as mouse clicks or interrupts including timer	Main loop, event handlers, asynchronous processes	Javascript, ActionScript, Visual Basic, Elm