1. Brief Introduction to Object-Oriented Languages

1.1 Object-Oriented vs Procedural Languages

Since the 1980s, there has been a great deal of interest in object-oriented programming (OOP).

- Early to mid 1950s:
  - Programs were written directly in machine language.
  - Assembler language (a symbolic form of machine language) was also used.
  - The programmer needed a good understanding of the architecture of the specific machine.
  - Programming even simple tasks was quite time taking.

- Late 1950s, early to mid 1960s:
  - First higher level languages (Fortran, Cobol, Algol) came into use.
  - Primary control structure was the goto statement.
  - Excessive use of goto could make even short code fragments very hard to understand (“spaghetti code”).

```c
/* Example in C. Actually, C hadn’t yet been developed. */
int n = 0;
x1: putchar('a');
   if ( n % 2 == 1 ) goto x3;
   if ( n / 2 == 1 ) goto x4;
   putchar('b');
x2: putchar('c');
x3: putchar('d');
   if ( n / 2 == 1 ) goto x5;
   putchar('e');
x4: putchar('f');
   ++n; goto x2;
x5: putchar('g');
```

- Late 1960s, 1970s:
  - Structured programming came into favor.
  - goto statements were replaced by control structures such as while and if/else.
  - Programs were subdivided into functions (or procedures).
    - Ideally, each function would be fairly short, and perform one specific task.
  - Among the languages for structured programming, introduced around 1970:
    - **Pascal**: developed by Niklaus Wirth.
    - **C**: developed by Dennis Ritchie.

- At first, programs were not subdivided into logical units, such as procedures.
- As programs increased in length, they became difficult to understand and debug.

- Late 1960s, 1970s:
  - Structured programming techniques were helpful for moderately large programs.
  - They were not completely adequate for very large programs:
    - Many programmers.
    - Possibly hundreds of thousands of lines of code (or even millions).
      - For example, Windows 2000: about 30,000,000 lines of code (?).
  - Structured programming was a step in the right direction, but more was needed.
  - Another development in the late 1960s:
    - A language named **Simula** was developed.
    - It was designed especially for programming simulations.
It was never as widely used as languages like Pascal and C.

Why mention it? It introduced some important concepts (e.g., classes) that later became key features of object-oriented programming.

It had a strong influence on C++.

1980s, early 1990s:

- **Object-oriented programming** came into favor.

- Object-oriented programming built upon (rather than replaced) structured programming.

1990s, early 2000s:

- Continuing interest in object-oriented programming, but with emphasis on building large programs out of **components** (possibly distributed components).

- The components are separate programs that
  - may be written in different languages, and
  - may execute on different machines (in the case of distributed components).

---

What is an object-oriented language?

- The term is not easy to define.

- We contrast object-oriented languages with **procedural languages**.

**Procedural languages** (e.g., Fortran, Pascal, Basic, C):

- Computation performed largely by means of assignment statements.

- Control structures (**if/else, while**, etc) used to vary the order in which assignment statements are performed.

- Heavy emphasis on procedures.

  - Procedures are active.

  - Data is passive.

  - Procedures act on data.

  - Data is passed to procedures, and returned by procedures.

---

**Object-oriented languages** (languages supporting OOP) were developed.

- The first true object-oriented language, Smalltalk, was developed in the late 1970s.

- Smalltalk, like C++, was strongly influenced by Simula.

**Late 1990s, early 2000s:**

- Component-based programming builds upon, rather than replaces, object-oriented programming.

  - Components communicate with each other through their **interfaces**.

  - Replacing an old component by a new one (supporting the same interface) has no effect on other components.

  - Component-based programming builds upon, rather than replaces, object-oriented programming.

  - With special library support and programming tools, object-oriented languages such as C++ or Java may be used to program components.
Object-oriented languages (e.g. Smalltalk, Eiffel, Java)

- Emphasis is on **objects** (data).
  - An object might be, for example,
    - a number,
    - a date,
    - a color,
    - a book,
    - a triangle,
    - a stack (say of integers),
    - a list (say of strings),
    - a window (on the screen),
    - an application (program),
    - etc.

- Objects are the active component.

- Each object contains internal data describing its **state**.
- In addition, each object contains **methods**.
  - These are routines that perform the operations appropriate for the object (possibly changing its state).
  - We compute with an object by sending **messages** to it.
    - Each message asks the object to invoke one of its methods.
- More details in the next section.

---

**Example** (Windows on a computer screen)

- **Object**: A specific window.
- **State**:
  - Position of window on screen.
  - Size of window.
  - Z-order (determines which window is displayed where several overlap).
  - etc.
- **Methods**:
  - Move the window.
  - Resize the window.
  - Change z-order.
  - Minimize / maximize the window.
  - etc.
- **Message to window** `w`, asking `w` to resize itself to 200 by 300 pixels.
  - `w.resize(200, 300);`