**COMPUTER SOFTWARE**

*Computer software* is a collection of related computer programs and their data. A *computer program* is a sequence of instructions written to perform some task, which a computing device can understand and execute. All computer programs share similar restrictions imposed on them by the nature of the computing hardware.

**Computer Program Organization**

A computer program makes the hardware usable by providing data and procedural abstractions for it.

![The Organization of a Computer Program](image)

Program data needed by the processing instructions as they execute are stored in the internal memory device. Long-term data are stored within external files on the external memory devices. Input instructions transfer data from an input device or an external file into the internal memory. Similarly, output instructions transfer data from the internal memory to an output device or an external file.

Some programs, such as a word processor, interact with a human user. Others, such as the cruise control on your automobile, interact with other devices. Still others, like a web browser, interact with a human on one end and a machine on the other.
**Program Data**
A computer program stores data into *variables*, each of which is referred to by an *identifier*.

**Example**
The Java statements below store values into two variables. The variable with identifier *radius* gets the value 6.25; that with identifier *diameter* gets 12.5.

```java
radius = 6.25;
diameter = 2 * radius;
```

A *variable* is a data abstraction representing a location in the internal memory where a datum is stored. Because of this, programmers often picture a variable as a box holding the datum.

**Example**
A conceptual picture of the variables *radius* and *diameter*.

<table>
<thead>
<tr>
<th>radius</th>
<th>diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>12.5</td>
</tr>
</tbody>
</table>

In the internal memory hardware, all data (e.g. text, music, images) are represented in sequences of zeroes and ones called *binary*. Each *binary digit* is called a *bit*, for short. There is nothing inherent in binary that indicates what kind of data (e.g. text, music, images) is represented.

**Example**
Consider the binary shown at right. If it represents a whole number then it is the value 65. If it represents text then it’s the letter ‘A’. If it represents a shade of the color green, then it’s . If it’s a gray level, then it’s .

To allow for the correct interpretation of binary, each variable must have a *data type*, which determines the kind of values that are stored into the variable.

**Example**
This Java statement specifies *miles* to be a variable with the data type *int*, which means that *miles* can only store whole numbers.

```java
int miles;
```
**Input Instructions**
Input instructions transfer data from an input device or an external file into the computer’s internal memory. Computer devices that perform input from a human user typically do so with *alphanumeric* characters (i.e. letters, digits and punctuation marks).

**Example**
When typed on the keyboard, 123.45 are collected as a sequence of alphanumeric characters:

```
1 2 3 . 4 5
```

Data in the computer’s internal memory are stored in binary. A computer program’s input instructions must include those that *scan* the input, converting the alphanumeric characters into binary.

```
1 2 3 . 4 5
```

```
100 0000 0101 1110 1101 1110
```
**Processing Instructions**

Processing instructions manipulate data stored in the internal memory device. Machine code instructions are limited operations such as moving data from one location to another or adding two data together. Computer languages such as Java, C++ or Python allow you to write statements that describe long sequences of machine operations.

<table>
<thead>
<tr>
<th>Java Statement</th>
<th>Machine Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius = 6.25;</td>
<td>(1) Move the constant 6.25 into internal memory and store in radius</td>
</tr>
<tr>
<td>diameter = 2 * radius;</td>
<td>(2) Move the constant 2 into the CPU’s arithmetic unit</td>
</tr>
<tr>
<td></td>
<td>(3) Move the value of radius from internal memory to the CPU’s arithmetic unit</td>
</tr>
<tr>
<td></td>
<td>(4) Multiply the two values</td>
</tr>
<tr>
<td></td>
<td>(5) Move the product from the CPU into internal memory and store in diameter</td>
</tr>
</tbody>
</table>

**Example**

Java statements and their possible equivalent machine operations.

**Output Instructions**

Output instructions transfer data from the internal memory to an output device or an external file. Computer output read by humans consists of alphanumeric data. In such cases, binary data in the computer’s internal memory must be converted into alphanumeric form prior to being sent to the output device.

**Sequential Execution**

Modern digital computers are designed according to an architecture devised by John von Neumann (1903–1957) in 1945.\(^1\) His design includes many of the features shown in the picture at the beginning of this topic, including separate central processor unit, internal memory and external memory. A key feature of von Neumann’s architecture is that machine instructions are stored within the internal memory, which are fetched and executed by a control unit that controls the flow of data through the central processing unit. Each machine instruction is fetched and executed in the order in which it appears in memory, a process called sequential execution. Consequently most computer languages adhere to this same process.

---

Example
As an example of sequential execution, consider the following Java code. Line 1 places the radius of a circle into the variable radius; line 2 calculates the area of the circle and stores it in area. The resulting memory appears as:

\[
\begin{array}{|c|}
\hline
\text{radius} & 10.0 \\
\text{area} & 314.159 \\
\hline
\end{array}
\]

Line 3 overwrites the contents of radius with a new value, 20. Memory appears as:

\[
\begin{array}{|c|}
\hline
\text{radius} & 20.0 \\
\text{area} & 314.159 \\
\hline
\end{array}
\]

The area is no longer correct with respect to the radius because the three statements are executed in sequential order; the area is not automatically recalculated.

1  radius = 10;
2  area = 3.14159 * radius * radius;
3  radius = 20;

Compare this behavior with that of a spreadsheet program (e.g. Microsoft Excel). When you enter a calculation into a spreadsheet program it establishes a relationship between the data that the program automatically maintains.

Example
A circle with radius \( r \) has area \( a = \pi r^2 \). This relationship applies to every circle. A spreadsheet that calculates the area of a circle is pictured below. When the user changes the radius, the spreadsheet automatically recalculates the area.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Radius:</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Area:</td>
<td>314.159</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Radius:</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Area:</td>
<td>1256.637</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>