Variables, Expressions and Selection Control

Symbols

In computing, a **symbol** is a set of one or more characters that represents something else.

For example, your name (all the letters together) is a symbol that represents you as an individual.

Literal Values

A **literal value** is a symbol that “stands for itself.” Thus,

1 is a symbol that represents the number one
10 is a symbol that represents the number ten

Literal values that represent numbers are called **numeric literals** (numeric values).

Numeric Literals

Numeric values can be **integer** values:

1 10 1024

or they can be **floating-point** (real) values:

1.0 1.20 10.245 1024.03525674

Arithmetic Expressions

Numeric values can be used in calculations. Such calculations are called **arithmetic expressions**.

In order to perform such calculations, a set of **arithmetic operators** is needed.

Typical Arithmetic Operators

Typical arithmetic operators in programming languages include:

+ (addition)
- (subtraction)
* (multiplication)
/ (division)

Examples of Arithmetic Expressions

2 + 4

2 * 4

(2 + 4) * 3

(2 + 4) * (6 – 3)

etc.

Note that if we don’t use parentheses, e.g., 2 + 4 * 3, then it is not clear whether the addition or the multiplication is done first.

Relational Operators

In programming, we often want to compare values. For example,

10 < 20 is true (less than operator)

The **relational operators** are,

< (less than), > (greater than)
==(equal to), != (not equal to)
<= (less than or equal to)
>= (greater than or equal to)

Boolean Operators

We can create larger true/false expressions by use of **Boolean (logical) operators**,

and
or
not

Example Use of Boolean Operators

(10 < 20) and (30 > 50) this expr is false

(10 < 20) or (30 > 50) this expr is true

10 < 20 is true

not (10 < 20) is false

Variables

Variables are indispensible in programming.

A **variable** is a symbol that stands for some literal value.

In programming, symbols that stand for something else are called **identifiers**.

Identifiers

Identifiers can include letters and digits, but must not start with a digit, for example,

sum	valid
totalSales2011	valid
2011TotalSales	invalid

The above style is called **camel case**, in which each word is capitalized, except for the first letter. The **underscore** is a special character that is often allowed, for example,

total_sales_2011

Example Variable Use

n = 10 (we say that n “holds” the value 10)

k = 5

n + 20 → 30

n + k → 15

n = n + 1 (if n is initially 10, it will become 11)

Initializing Variables

n = 10   (we say that n “holds” the value 10)

n + 20 → 30   OK
n + k → 15   INVALID – variable k not assigned a value

So what if we wanted to add up values, the result stored in variable sum.

n = n + 1   (if n is initially 10, it will become 11)

Initializing Variables (cont.)

```
sum = sum + 10
sum = sum + 20
sum = sum + 30
```

What is the final value in sum?

Initializing Variables (cont.)

The answer is, we can’t know. The reason is that we don’t know what the initial value of sum may be,

```
sum = sum + 10
sum = sum + 20
sum = sum + 30
```

If sum had an initial value of 0, the answer would be 60. But what if its initial value were 5? 10? etc.

Initializing Variables (cont.)

This example demonstrates why it is important to initialize variables *if they are going to be used before they have been assigned*,

```
sum = 0            initialization
sum = sum + 10     without the initialization, variable
                   sum would be used here containing
                   some unknown value
sum = sum + 20
sum = sum + 30
```

Now we can be guaranteed to get the right result (60).

Control

Three forms of control:

- Sequence
- Selection
- Repetition (iteration)

Sequence control is when instructions are executed in the order that they are listed.

Selection Control

Selection control is used to select among two or more set of instructions to execute, based on given conditions.

A **condition** is any true/false (Boolean) expression.

Example Selection Control

“If it is raining outside, then I will take my umbrella. Otherwise, I will not take it.”

In programming, we would structure this statement as,

```
if raining today
   take umbrella
else
   don’t take umbrella
```

Selection Control with no Else

Selection control is not only used to select among two sets of instructions, but is also used to either do or not do a single set of instructions,

```
if raining today
  take umbrella
```

In this case, the “else” section is omitted.

Chained Selection Control

Selection control instructions can be **chained**, e.g.,

```
if grade on exam is >= 90
    grade is an A
else
if grade on exam is >= 80
    grade is a B
else
if grade on exam is >= 70
    grade is a C
else
if grade on exam is >= 60
    grade is a D
else
    grade is an F
```

*This called the **“catch all” case**.*
The catch-all case is optional.

Nested Selection Control

Selection control instructions can be **nested**, e.g.,

```
if grade on exam is >= 90
    if grade >= 97
        grade is A+
    else
    if grade >= 93
        grade is A
    else
    if grade >= 90
        grade is an A-
else
if grade on exam is >= 80
    if grade >= 87
        grade is B+
etc.
```