Type-Casting

Things That You Should Know

- Primitive C# data types
- Declaring variables and object
- Basic C# assignment and mathematical operators

Lecture Outline

- Expressions
- Type-casting
  - Implicit
  - Explicit
    - Numerical type-casting
    - Type-casting from numbers to strings
    - Type-casting from strings to numbers

Expressions

- Expressions are operational statements that requires further evaluation.

```
decimal d2rcf =  3.14159 / 180.0;
```

- The underlined segment is a mathematical expression.

Expressions

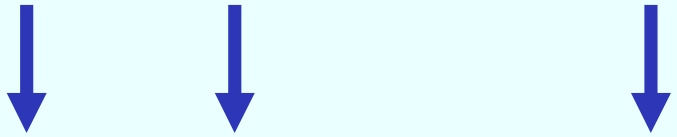
- Data type compatibilities can become an issue when working with expressions
- Data types must be considered when constructing a complicated expression such as:

```
float x = xp + h * Math.Cos(angle);
```

Expressions

- The assignment statement below must be carefully constructed ensure data type operational compatibility.

```
float x = xp + h * Math.Cos(angle);
```



What are the involved data types?

int, float, double, long, etc.

Expressions

- The general rule for constructing expressions is to use operands with similar data types.
- However, this is not always possible and so C# allows us to convert from 1 numerical data type to another.

Type-Casting

- The processes of converting data values from one data type to another.
- Type-casting may be categorized as:
  - Implicit
  - Explicit

Implicit Type-Casting

- This method of type-casting is automatically done by C# and is carried out only when there is no risk of data loss.

Samples of Implicit Type-Casting

Sample Set 1: Conversion from smaller to larger data type capacity

```
double a = 3.14159f;

short b = 10;
int c = b;

float e = 1.12548f;
double f = e;
```

Samples of Implicit Type-Casting

Sample Set 2: Conversion from simple to more complex numerical data types

```
float x = 10;

int y = 300;
double z = y;
```

Explicit Type-Casting

- This type of numerical conversions are needed in scenarios where possible loss of data may result from the data conversion.
- Extra care must be taken when performing this type-casting since C# will assume that the loss of data is accounted for by the programmer

Explicit Type-Casting

- Explicit type-casting means we'll need to write codes to tell C# that we are forcing a data conversion regardless of the risk of data loss.
- From this point on, we will use the term type-casting synonymously with explicit type-casting since we will never need to write additional codes to perform implicit type-casting

Type-Casting

- To convert from one numerical data type to another enclose the data type that a data value will be converted to in a set of parenthesis and place this on the left side of the item to be converted, see example below.

```
float a = (float)3.14159;
long b = (long)a;
```

Type-Casting

- Generally, explicit type-casting is needed when we like to convert values of data type with larger capacity to one with a smaller capacity

```
int a = 15000;
short b = (short)a;
```

- This is necessary because C# is not sure of what values users may want to put into the variable ***a*** or if the values can fit in the smaller data type short (There is a risk of data loss).

Type-Casting

- Explicit type-casting is also needed when converting values from complex data types to more simple data types.

```
float a = 3.14159f;
int b = (int)a;
```

- In this case, assigning the value of ***a*** to the variable ***b*** will definitely result to loss of data since the value for decimal places is totally discarded when ***a*** is assign to ***b***.

Type-Casting

- When operational expressions are used for computational purposes. It is important to understand that some operations might change the resulting data type of the entire expresssion.

```
int a = (int)(10 / 3);
```

- Something like this might cause an error even though both 10 and 3 are integers, the division operator results to a number with decimal values and so type-casting is also necessary to solve such a computation

Type-Casting

- Special care must be taken when constructing complex mathematical expressions so that the proper type-cast is applied appropriately.

End


DE LA SALLE-COLLEGE OF SAINT BENILDE
INFORMATION SYSTEMS PROGRAM