# Structure of C# Programs and Identifier Naming

## Identifier naming rules in C#

• A name must consist of only letters (A-Z,a-z), digits (0-9), or underscores ( )

• The first character must be either a letter or an underscore

• A name can be at most 63 characters in length

• A name must not be identical to a reserved word such as class, namespace, int, void, static

### 1.1: Identifier Naming

Consider the following names and check if each of them is a valid name for a C# identifier.

XXX

\_Y

$$$ string

i.j

Student ID

HelloWorld!!

first-time

### 1.2: Basic Structure of C# Programs

Here is an example of C# program structure.

namespace \_\_\_ ( A ) \_\_\_ {

class \_\_\_ ( B ) \_\_\_ {

static void Main () {

\_\_\_ ( C ) \_\_\_

}

}

}

From the above, the positions (A), (B), and (C) have the following meanings:

(A) is for the namespace’s name

(B) is for the class’s name

(C) is for statements telling the computer what to do

Now complete the following tasks:

* Write a program that compiles and runs. Your program must meet these requirements:
  + namespace has the name TestingNamespace
  + class has the name TestingClass
  + The program outputs no results
* Write another program that can also compile and run, but now it must meet these requirements:
  + Shares the namespace with the previous program
  + class has the name SayHello
  + The program outputs a phrase “Hello”

# Data Types and Variables

## Data types in C#

Type Description

|  |  |
| --- | --- |
| char | Single character |
| bool | Truth value – true or false |
| byte | Unsigned integer between 0 and 255 |
| int | Signed integer between -2,147,483,648 and 2,147,483,647 |
| uint | Unsigned integer between 0 and 4,294,967,295 |
| long | Signed integer between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 |
| ulong | Unsigned integer between 0 and 18446744073709551615 |
| float | Real number |
| double | Double-precision real number |
| string | Sequence of characters |

## Declaration of Variables and Constants

### Variable Declaration

Declaring the variable myvar to be of type uint:

uint myvar;

Declaring the variable myvar to be of type long and have the initial value 30000:

long myvar = 30000;

### Constant Declaration

Declaring the constant myconst to be of type double and have the fixed value 2.717:

const double myconst = 2.717;

Examples:

const float PI = 3.414;

const int FreezingPoint = 32;

int X, Y;

int AREA;

char ch = ’A’;

string mynote = "Hello, Kitty";

int j = 5;

### 2.1: Variable and Constant Declaration

Type the following code

using System ;

class ComputeArea {

static void Main () {

const double PI = 3.1415926535;

radius = 12.5;

area = PI \* radius \* radius ;

Console . WriteLine ( " Circle area = {0} " , area );

}

}

Then try to compile the program. What errors are reported, and at which lines? What do you think is the cause?

Fix the above code so that it compiles and runs. How did you fix the errors?

### 2.2: Variable Declaration (2)

Type the following code

using System ;

class mod2lab2 {

static void Main () {

int f = 1.5 F ;

char s = " Harry Potter " ;

char c = ’A ’;

Console . WriteLine ( " f value = {0} , s = {1} , c = {2} " ,

f , s , c );

}

}

Fix the above code so that it compiles and runs. How did you fix the errors?

### 2.3: Choosing Data Types

For each of the data item given below, choose an appropriate data type and write a C# statement to declare the variable:

* Variable myAge to store your own age
* Variable income to keep track of Peter’s personal income
* Variable temp c to store temperature in degree Celcius
* Variable temp k to store temperature in Kelvin
* Variable name to store Aum’s full name

# Mathematic Expressions

Mathematic operators in C# are as follows:

|  |  |
| --- | --- |
| Operator | Symbol |
| Add | + |
| Subtract | – |
| Multiply | \* |
| Divide | / |
| Modulo | (division remainder) % |
| Grouping | ( ) |

Type the following code

using System ;

class Test {

static void Main () {

double x = 3.0 , y = 2.0;

int a = 10 , b = 2;

Console . ReadLine ();

}

}

Notice that line 6 is blank. Now fill the blank with each of the statements listed in the table below, one at a time. Run the program and put the result in the right column.

|  |  |
| --- | --- |
| Statement | Result |
| Console.WriteLine(a) | 10 |
| Console.WriteLine(x+a) |  |
| Console.WriteLine(a/b) |  |
| Console.WriteLine(y/x) |  |
| Console.WriteLine(y%x) |  |
| Console.WriteLine((a+b)/b%a) |  |
| Console.WriteLine(9.0/5.0\*(a-x)) |  |
| Console.WriteLine(x+y-x\*y%x) |  |
| Console.WriteLine(57%50/25) |  |

# Output Statements

### 4.1 Basic usage of Console.Write and Console.WriteLine

Write a C# program that outputs the following result via by using a single WriteLine method:

Live as if you were to die tomorrow.

Learn as if you were to live forever.

-Mahatma Gandhi

### 4.2 Advanced usage of Console.Write and Console.WriteLine

Consider the following incomplete code:

using System ;

class SayHi {

static void Main () {

string yourName = \_\_\_ (1) \_\_\_ ;

uint yourAge = \_\_\_ (2) \_\_\_ ;

Console . WriteLine ( " Hello {1}. You are {0} years old . " ,

\_\_\_ (3) \_\_\_ , \_\_\_ (4) \_\_\_ );

}

}

Fill in the blanks (1),..., (4) so that the program will say hello to you and print out your own age. For example, if your name is “Arthur” and you are 18 years old, the result should be: Hello Arthur. You are 18 years old.