Programming in C   I

Content

Introduction

- A computer program
  - A **computer program** is a sequence or set of instructions in a *programming language* for a computer to execute.
  - A **computer** is a machine that performs computations based on instructions.
  - A computer is made up of two components:
    - Software and Hardware
  - **Software** consists of a set of instructions for the hardware.
    - These instructions are typically saved in files on your computer.
    - These instructions are in a special type of language, called a ***programming language***.

Introduction

- A computer program
  - A computer program in its human-readable form is called **source code**
  - Source code needs another computer program to execute because computers can only execute their native **machine instructions *(in machine language)***
  - Therefore, source code may be translated to machine instructions using the language's **compiler**
  - A **C compiler** translates the source code into machine language
  - The most frequently used C compiler is the GNU C/C++ compiler

Introduction

- Computer language
  - The native language of a computer is binary ***that is ones and zeros*** which is called ***machine language***
  - The earliest digital electronic computers were programmed directly in binary which is is tedious, complex and error-prone
  - For example: Suppose you want to write a program to calculate the area of a rectangle using machine language.

    Formula: area = length * width
  - To write the above program, you might need the following sequence of instructions.

    100100 010001

    100110 010010

Introduction

- Assembly languages
  - The programmer had to remember the machine language codes for various operations and locations of the data in the main memory, which is really difficult
  - **Assembly languages** were developed to make the programmer's job easier though they were not also easy to use but easier than machine language
  - A program called **an assembler** translates the assembly language instructions into machine language.
  - Both machine language and assembly language are also called **low-level programming languages**

Introduction

- High-level programming languages
  - High level programming languages are very closer to natural languages, such as English or Swahili
  - These languages were made to make programming work easier than using assembly languages or machine language.
  - Examples of high level programming languages may include:

FORTRAN, COBOL, Pascal, C, C++, C#, Java, Java Script, etc.

Basics of C Programming

- Introduction
  - C is a high level programming language
  - C is a general-purpose programming language, and is used for writing programs in many different domains, such as operating systems, numerical computing, graphical applications, etc.
  - C language consist of some characters set, numbers and some special symbols.
  - The character set of C consist of all the alphabets of English language. C consist of Alphabets a to z, A to Z ; Numeric 0,1 to 9; Special Symbols {,},[,],?,+,-,*,/,%,!,;,and more

Basics of C Programming

- Introduction
  - A sample C program

```
#include <stdio.h>
main()
{
/* My first program */
printf("Hello World! \n");
}
```

Basics of C Programming

- Introduction
  - A sample C program
    - The C program starting point is identified by the word **`main()`**
    - This informs the computer as to where the program actually starts
    - The parentheses that follow the keyword main indicate that there are no arguments supplied to this program
    - The two braces, ***{*** and ***}***, signify the begin and end segments of the program
    - Braces are used throughout C to enclose a block of statements to be treated as a unit.

Basics of C Programming

- Introduction
  - A sample C program
    - The first line ***`#include <stdio.h>`*** is a *pre-processor command*.
    - The pre-processor command tells a C compiler to include *stdio.h header file* before going to actual compilation.
    - The line ***`/* My first program */`*** shows the use of comments in C. It can also be presented as ***`//My first program`***

Basics of C Programming

- Introduction
  - A sample C program
    - `printf()` is actually a function (procedure) in C that is used for printing variables and text.
    - The text appears in double quotes **" "**, is printed without modification (some exceptions exists for modification).

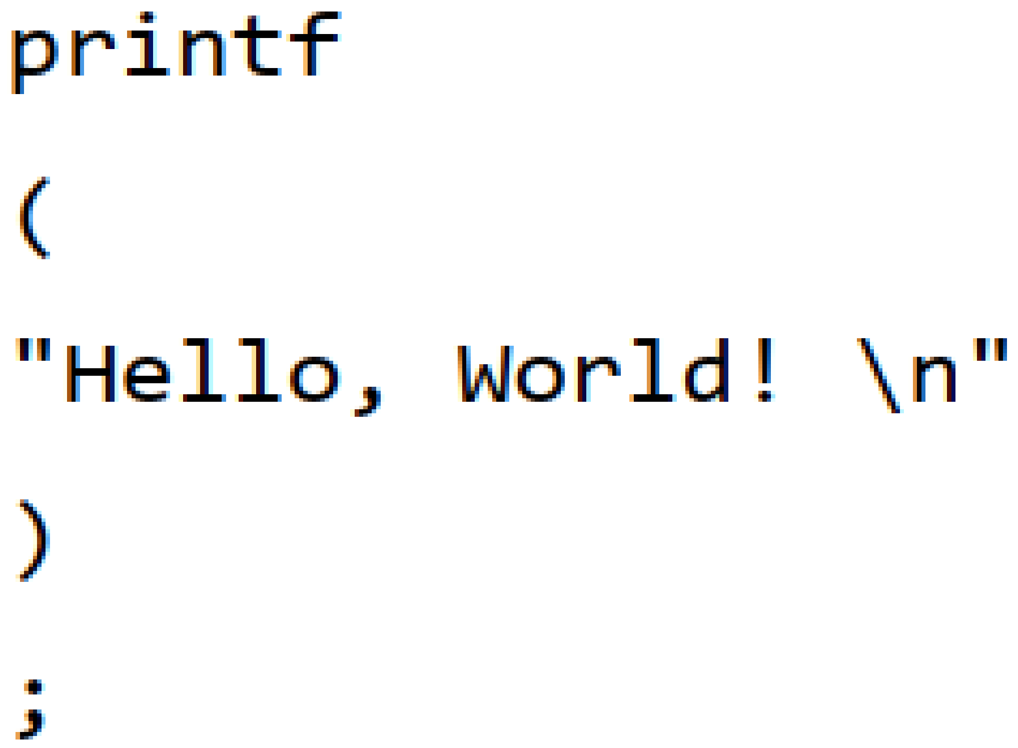
Basics of C Programming

- Basic Concepts
  - The words formed from the character set of C are the building blocks of C and are sometimes known as **tokens**
  - These tokens represent the individual entity of the language
  - A token is either **a keyword, an identifier, a constant, a string literal, or a symbol.**
  - For example, the following statement consists of five tokens:

printf ("Hello, World! \n");

Basics of C Programming

- Basic Concepts
  - The individual tokens are as shown:



```
printf

(

"Hello, World! \n"

)

;
```

Basics of C Programming

- Basic Concepts
  - **Identifiers**
    - Identifiers are nothing but the names given to the elements in a C program.
    - A C program consist of two types of elements, user defined and system defined.
    - An identifier is a word used by a programmer to name a variable, function, or constants.
    - Identifiers in C must begin with a character or underscore, and may be followed by any combination of characters, underscores, or the digits 0-9.
    - Both Upper and lowercase letters can be used

Basics of C Programming

- Basic Concepts
  - **Identifiers**
    - C is case sensitive
    - Uppercase and lowercase letters are considered different
    - Thus, the identifier **a_Number** is not the same as the identifier **a_number**
    - Identifiers should be *meaningful (but short)* names
    - **Keywords** are reserved identifiers that have strict meaning to the C compiler
    - Example of reserved words:

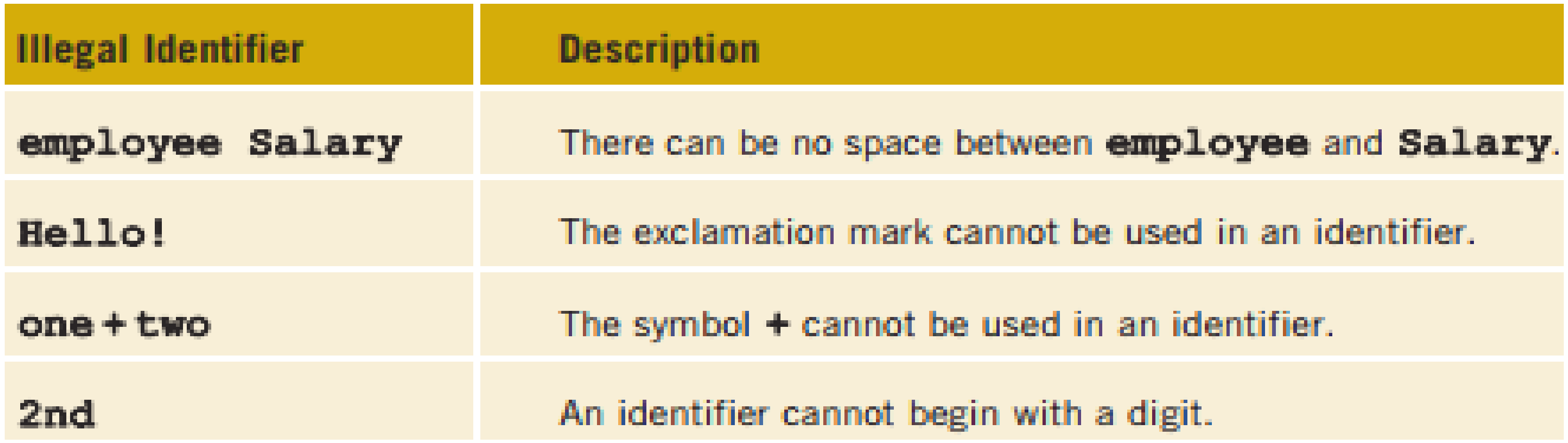
```
if, else, char, int, while
```

Basics of C Programming

- Basic Concepts
  - Identifiers
    - **Example of legal identifiers**

      `Summary,` `exit_flag,` `i,` `Joe7,` `Number_of_moves,` `_id`

    - **Example of illegal identifiers**



Illegal Identifier	Description
`employee Salary`	There can be no space between **employee** and **Salary**.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Basics of C Programming

- Basic Concepts
  - **Data Types**
    - Data type refers to a set of values together with a set of operations
    - Basic data type includes:
      - Integer (int)
      - Float (float)
      - Character (char)

Basics of C Programming

- Basic Concepts
  - **Data Types - Integer**
    - Are numbers such as the following:

    -6745, -65, 0, 73, 36742, +723
    - Positive integers do not need a + sign in front of them
    - No commas are used within an integer
    - The keyword used to define integers is **`int`**

Basics of C Programming

- Basic Concepts
  - **Data Types - *Float***
    - Deals with decimal numbers
    - Floating data type has two categories:
      - Float
      - Double
    - **Float data** type is used in C to represent any decimal number between -3.4 *1038 and 3.4 *1038.
    - The keyword used to define float is `float`
    - **Double data** type is used in C to represent any decimal number between -1.7 *10308 and 1.7 *10308.
    - The keyword used to define double is `double`

Basics of C Programming

- Basic Concepts
  - **Data Types - *Float***
    - The maximum number of significant digits is called the precision
    - Float values are called *single precision*, and values of type double are called *double precision*
    - If you need accuracy to more than six or seven decimal places, you can use the double type

Basics of C Programming

- Basic Concepts
  - **Data Types - *Character***
    - **Character** in C, represent single characters such as letters, digits, and special symbols
    - Examples of values belonging to the char data type include the following:

      'A', 'a', '0', '*', '+', '$', '&', ' ‘
    - The data type char allows only one symbol to be placed between the single quotation marks
      - Example: The value 'abc' is not of the type char
    - The keyword used to define character is **`char`**

Basics of C Programming

- Basic Concepts
  - **Variables**
    - **A variable** is a named memory location in which data of a certain type can be stored
    - The contents of a variable can change
    - User defined variables must be declared before they can be used in a program
    - A syntax rule to declare a variable is:

      dataType identifier;
    - Variable declaration example:

```
int xy; double x; char letter;
```

Basics of C Programming

- Basic Concepts
  - **Variables**
    - Multiple variables of same data type can be declared as:

      dataType identifier1, identifier2, identifier3, ….. ;
    - Multiple variables of different data type can be declared as:

      dataType1 identifier1; dataType2 identifier2;

      dataType3 identifier3; …….
    - **Example**

```
int xy;
Double c; char grade;
```

Basics of C Programming

- Basic Concepts
  - **Constants**
    - A constant refers to a memory location whose content is not allowed to change during program execution.
    - A syntax rule to declare a constant is:

      const dataType identifier = value;
    - Examples of constants declaration

      const int students = 20;

      const double conv = 2.54;

Basics of C Programming

- Basic Concepts
  - **Expression**
    - An **expression** in C, is some combination of constants, variables, operators and function calls.
    - Sample expressions are:

```
a + b
3.0 * x – 9.66553
tan(angle)
t = u + v
x <= y
++j
```

Basics of C Programming

- Basic Concepts
  - **Statement**
    - A **statement** in C is just an expression terminated with a semicolon. For example:

```
sum = x + y + z;
printf("Hello World!");
```

    - Types of statements:
      - Expression statements
      - Compound statements
      - Control statements

Basics of C Programming

- Basic Concepts
  - **Expression Statement**
    - Consists of an expression followed by a semicolon
    - The execution of such a statement causes the associated expression to be evaluated
    - For example:

```
a = 6;
c = a + b;
++j;
```

Basics of C Programming

- Basic Concepts
  - **Compound Statement**
    - Consists of several individual statements enclosed within a pair of braces { }
    - The individual statements may be expression statements, compound statements, or control statements.
    - Unlike expression statements, compound statements do not end with semicolons
    - For example:

```
{
pi = 3.141593;
circumference = 2. * pi * radius;
area = pi * radius * radius;
}
```

Basics of C Programming

- Basic Concepts
  - **Control Statement**
    - Consists of a selection statement whereby an action is executed from two or more options
    - For example:

```
if(age<18){
   printf(“You can not vote");
}else{
   printf(“You can vote");
}
```

Basics of C Programming

- Basic Concepts
  - **Assignment operator**
    - Is the equal sign = used to give a variable the value of an expression. For example:

```
x=34.8;
sum=a+b;
slope=tan(rise/run);
midinit='J';
j=j+3;
```

Basics of C Programming

- Basic Concepts
  - **Assignment operator**
    - As an assignment operator, the equal sign should be read as *“gets”*
    - In the assignment statement a=7; two things actually occur. The integer variable **a** gets the value of **7**, and the expression **a=7** evaluates to **7**.

Basics of C Programming

- Basic Concepts
  - **Variable initialization**
    - A variable initialization refers to the first time a value is placed in the variable
    - C Variables may be initialized with a value when they are declared. For example:

```
int x = 3;
```

Basics of C Programming

- Basic Concepts
  - **Arithmetic Operators**
    - There are six primary arithmetic operators in C:
      - Negation (-)
      - Modulus (%)
      - Multiplication (*)
      - Addition (+)
      - Division (/)
      - Subtraction (-)

Basics of C Programming

- Basic Concepts
  - **Arithmetic Operators**
    - The operators work as follows:
    - Use the operators +, -, *, and / with both integer and floating point data types
    - Use % with only the integer data type, to find the remainder in ordinary division
    - Using / with the integer data type, it gives the quotient in ordinary division
    - Integer division truncates any fractional part; there is no rounding

Basics of C Programming

- Basic Concepts
  - **Arithmetic Expressions**
    - An arithmetic expression refers to an expression that contains operator(s) and operand(s). Example:

      -5, 8 – 7, 3 + 4, 2 + 3 * 5, 5.6 + 6.2 * 3

      x + 2 * 5 + 6 / y, where x and y are unknown numbers
    - The numbers appearing in the expressions are called **operands**
    - The numbers that are used to evaluate an operator are called the operands for that operator.

Basics of C Programming

- Basic Concepts
  - **Arithmetic Expressions**
    - Three types of arithmetic expressions in C:
      - **Integer expressions** - all operands in the expression are integers.
      - An integer expression yields an integer result.
      - **Floating-point (decimal) expressions** - all operands in the expression are decimal numbers.
      - A floating-point expression yields a floating-point result.
      - **Mixed expressions** - the expression contains both integers and decimal numbers.

Basics of C Programming

- Basic Concepts
  - **Arithmetic Expressions**
    - Integer expressions

      2 + 3 * 5

      3 + x - y / 7
    - Floating-point (decimal) expressions

      12.8 * 17.5 - 34.50
    - Mixed expressions

      6 / 4 + 3.9

      5.4 * 2 - 13.6 + 18 / 2

Basics of C Programming

- Basic Concepts
  - **Mixed Expressions evaluation rules**
    - When evaluating an operator in a mixed expression the following rules apply:
      - If the operator has the same types of operands, the operator is evaluated according to the type of the operands
      - If the operator has both types of operands, the integer is changed to a floating-point number with the decimal part of zero and the operator is evaluated
    - The entire expression is evaluated according to the precedence rules

Basics of C Programming

- Basic Concepts
  - **Order of Precedence**
    - In expressions that have more than one arithmetic operator, the expression is evaluated using operator precedence rules
    - According to the order of precedence rules for arithmetic, *, /, % are at a higher level of precedence than +, -
    - The operators *, /, and % have the same level of precedence
    - The operators + and - have the same level of precedence

Basics of C Programming

- Basic Concepts
  - **Order of Precedence**
    - When operators have the same level of precedence, the operations are performed from left to right.
    - Example, evaluate the following expression using order of precedence

```
3 * 7 – 6 + 2 * 5 / 4 + 6
Solution
= 3 * 7 – 6 + 2 * 5/ 4  + 6
= 21 – 6 + 10 / 4 + 6 (Evaluate *)
= 21 – 6 + 2 + 6 (Evaluate /.)
= 15 + 2 + 6 (Evaluate –)
= 17 + 6 (Evaluate first +)
= 23 (Evaluate +)
```

Basics of C Programming

- Basic Concepts
  - **Assignment Statements**
    - Simple assignment statements.

      Example

      ```
      int x;
      x = 5;
      ```
    - Compound assignment statements.
      - Corresponding to the five arithmetic operators +, -, *, /, and %; C provides five compound operators: +=, -=, *=, /=, and %=, respectively

Basics of C Programming

- Basic Concepts
  - **Assignment Statements**
    - Compound assignment statements
      - Consider the following simple assignment statement, in which x and y are int variables:

        x = x * y;
      - Using the compound operator *=, this statement can be written as:

        x *= y;
      - Using the compound operator *=, you can rewrite the simple assignment statement:

        variable = variable * expression;

      as:

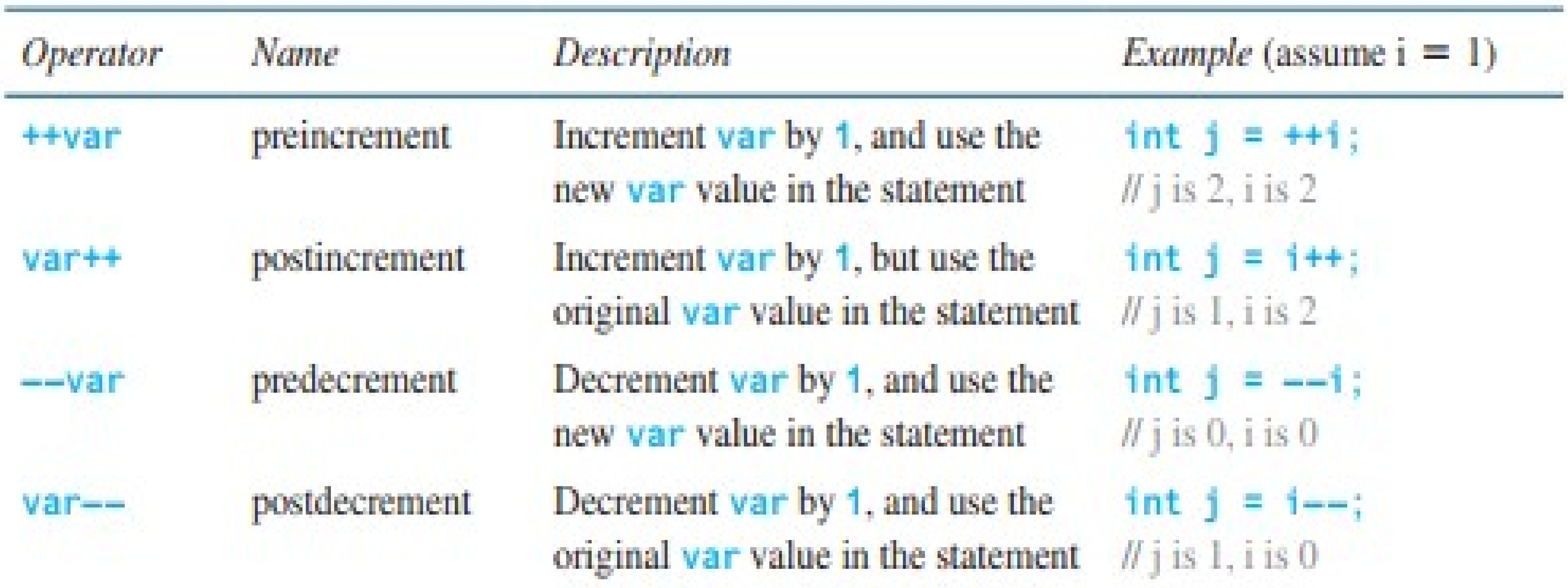
        variable *= expression;

Basics of C Programming

- Basic Concepts
  - **Increment and Decrement Operators**
    - The increment operator (++) and decrement operator (- -) are for incrementing and decrementing a variable by 1
    - The syntax of the increment operator is:
      - Pre-increment: **`++variable`**
      - Post-increment: **`variable++`**
    - The syntax of the decrement operator is:
      - Pre-decrement: **`– –variable`**
      - Post-decrement: **`variable– –`**

Basics of C Programming

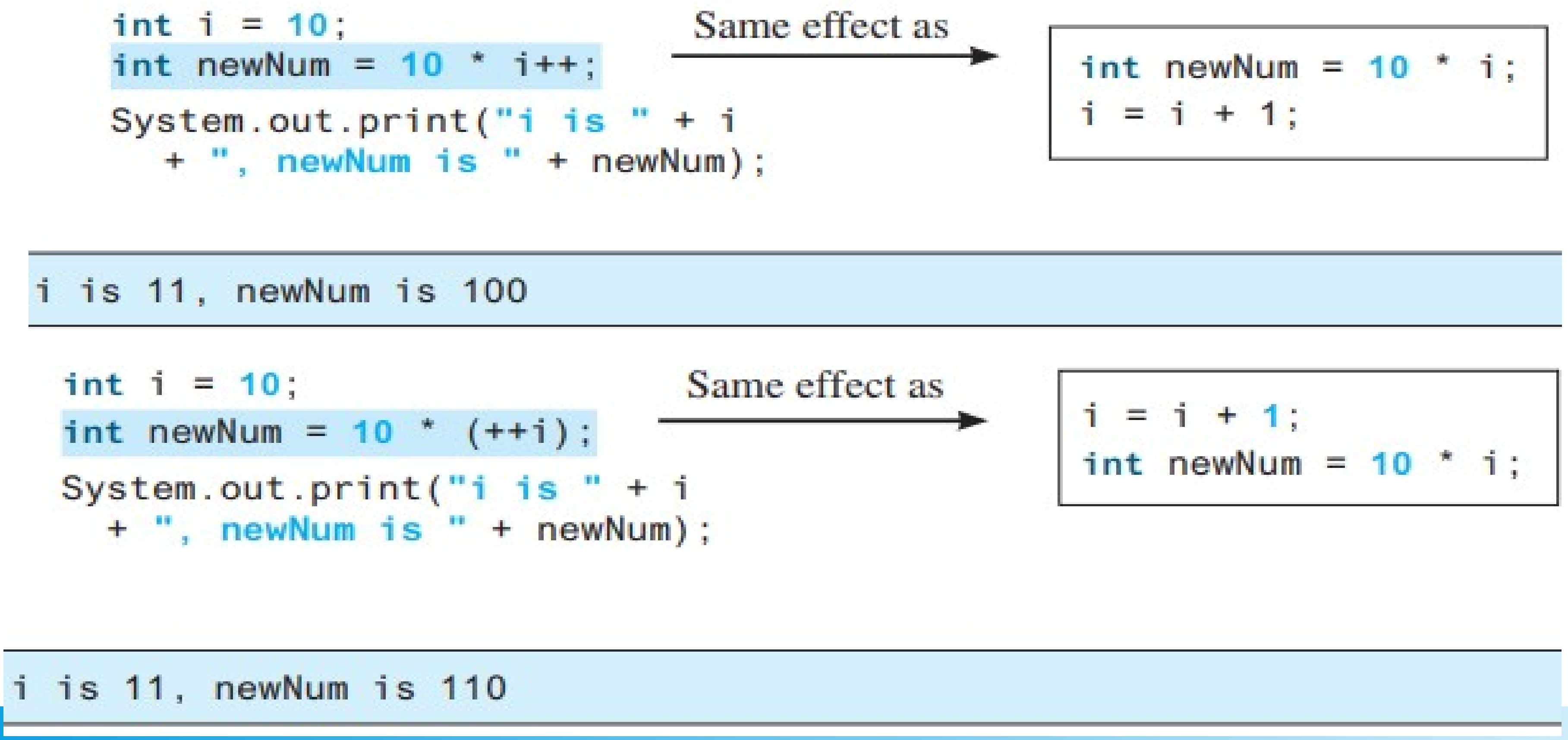
- Basic Concepts
  - **Increment and Decrement Operators**



Operator	*Name*	*Description*	*Example* (assume i = 1)
`++var`	preincrement	Increment var by 1, and use the new var value in the statement	`int j = ++i;` // j is 2, i is 2
`var++`	postincrement	Increment var by 1, but use the original var value in the statement	`int j = i++;` // j is 1, i is 2
`--var`	predecrement	Decrement var by 1, and use the new var value in the statement	`int j = --i;` // j is 0, i is 0
`var--`	postdecrement	Decrement var by 1, and use the original var value in the statement	`int j = i--;` // j is 1, i is 0

Basics of C Programming

- Basic Concepts
  - **Increment and Decrement Operators**
    - **Example**


Same effect as
Same effect as

```
int i = 10;
int newNum = 10 * i++;
System.out.print("i is " + i
   + ", newNum is " + newNum);
```

```
int newNum = 10 * i;
i = i + 1;
```

```
i is 11, newNum is 100
```

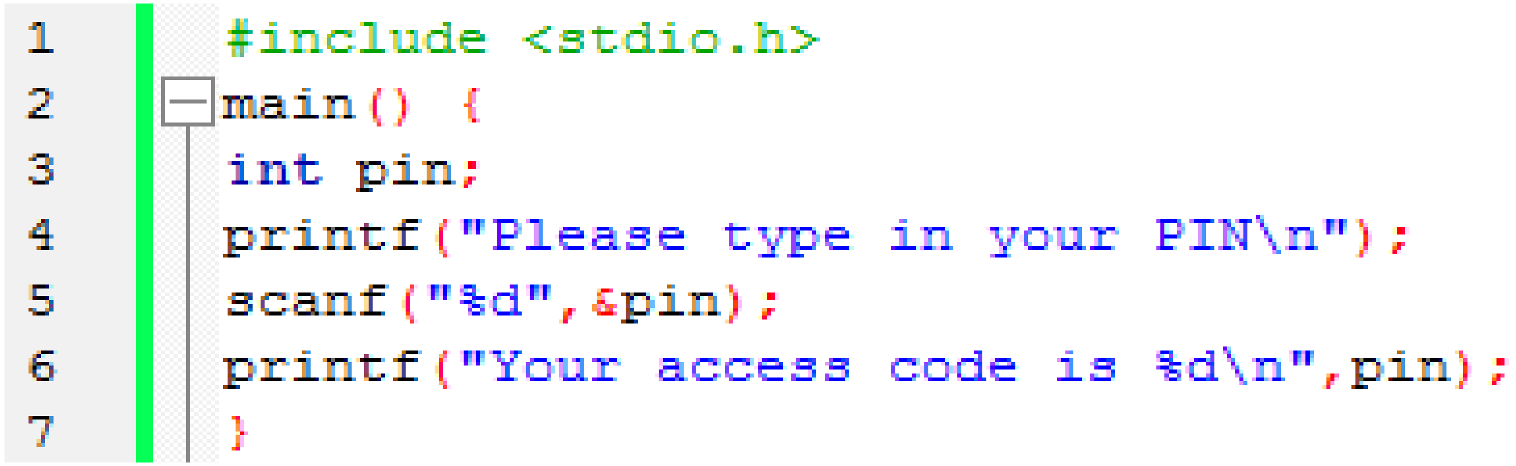
```
int i = 10;
int newNum = 10 * (++i);
System.out.print("i is " + i
  + ", newNum is " + newNum);
```

```
i = i + 1;
int newNum = 10 * i;
```

```
i is 11, newNum is 110
```

Basics of C Programming

- Basic Concepts
  - **Basic Input**
    - Inputs in C are done by using the scanf function
    - Consider the example below



```c
#include <stdio.h>
main()  {
int pin;
printf("Please type in your PIN\n");
scanf("%d",&pin);
printf("Your access code is %d\n",pin);
}
```

Basics of C Programming

- Basic Concepts
  - **Basic Input**
    - Line 5 shows how the scanf function can be used
    - The scanf function has a control string and an address list
    - The **&pin** specifies the memory location of the variable the input will be placed in

Basics of C Programming

- Basic Concepts
  - **Basic Output**
    - Consider the C program example below:

```
int sum = 33;
printf(“value of sum is %d\n”, sum);
```

  - The second statement will produce the following output:

```
value of sum is 33
```

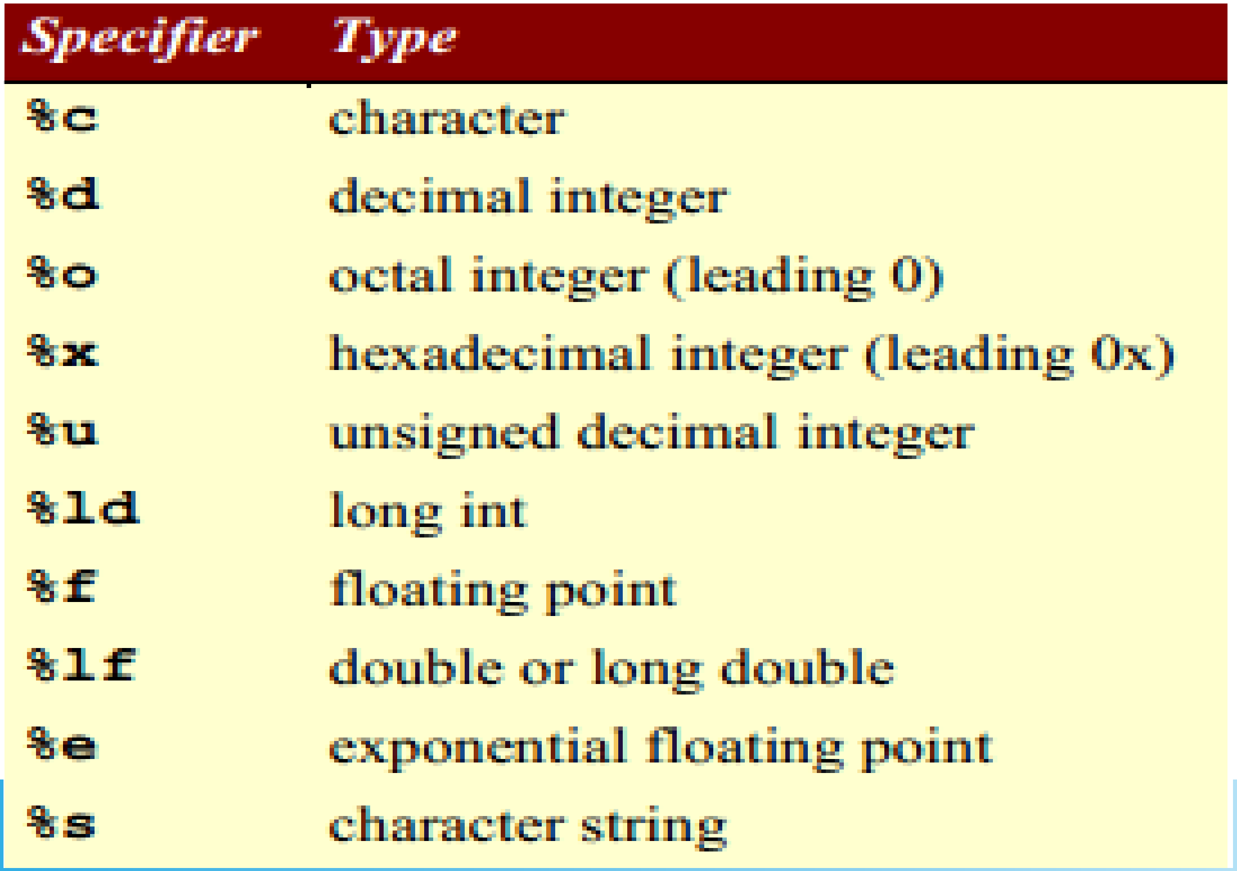
  - The first argument of printf function is called the control string.

Basics of C Programming

- Basic Concepts
  - **Basic Output**
    - The **%** sign is a special character in C and marks the beginning of a format specifier
    - A format specifier controls how the value of a variable will be displayed on the screen
    - The character **d** that follows indicates that a decimal integer will be displayed
    - The **\n** is a special character for printing a new line.

Basics of C Programming

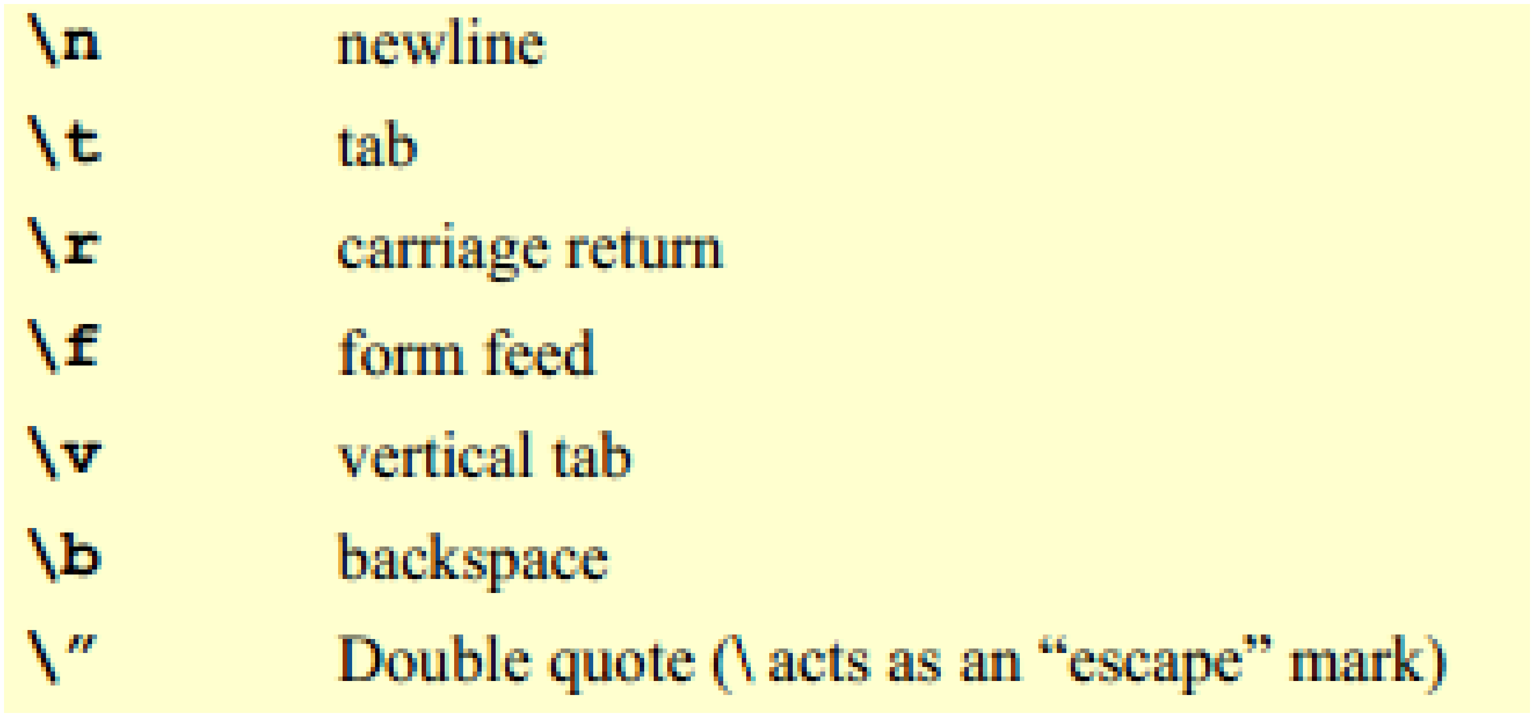
- Basic Concepts
  - **Basic Output**
    - The format specifiers with their data types



Specifier	*Type*
`%c`	character
`%d`	decimal integer
`%o`	octal integer (leading 0)
`%x`	hexadecimal integer (leading 0x)
`%u`	unsigned decimal integer
`%ld`	long int
`%f`	floating point
`%lf`	double or long double
`%e`	exponential floating point
`%s`	character string

Basics of C Programming

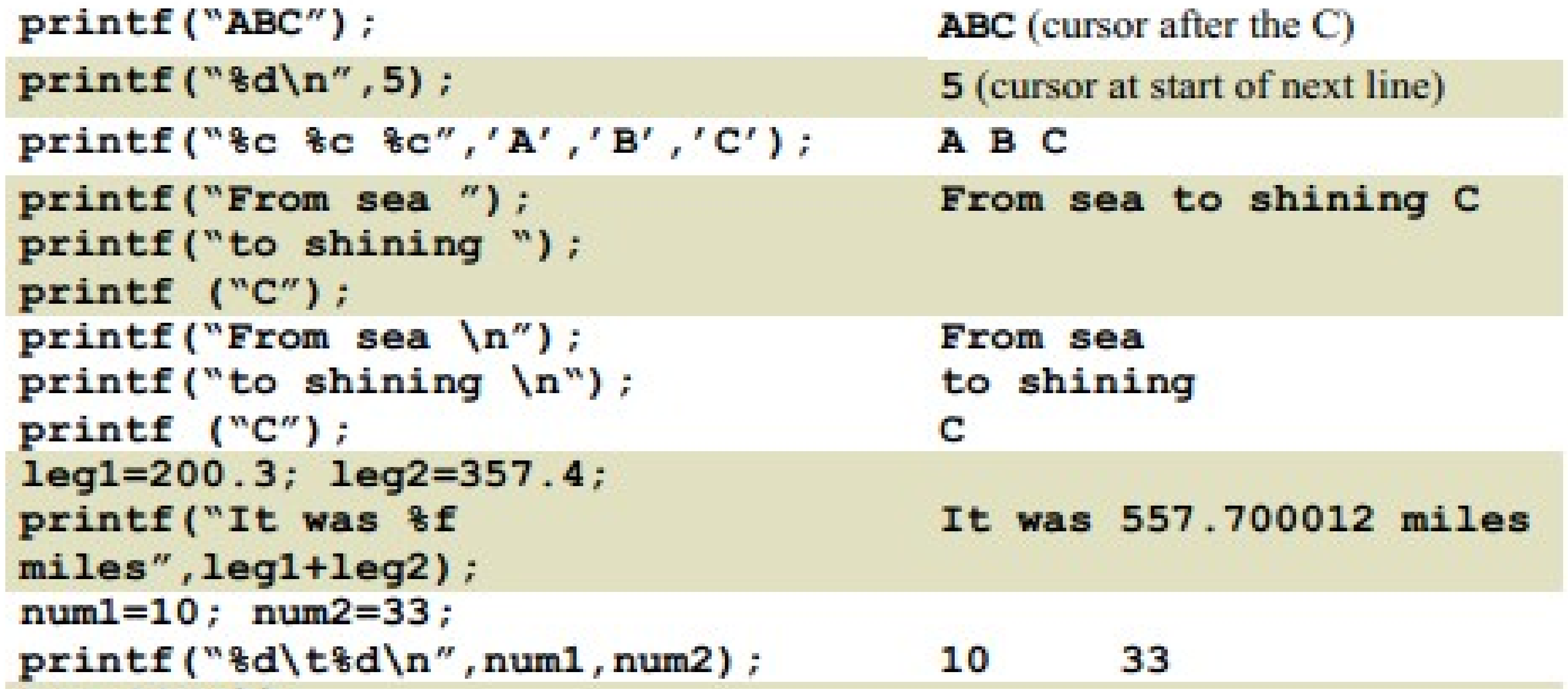
- Basic Concepts
  - **Basic Output**
    - Some special character for cursor control



`\n`	newline
`\t`	tab
`\r`	carriage return
`\f`	form feed
`\v`	vertical tab
`\b`	backspace
`\"`	Double quote (\ acts as an “escape” mark)

Basics of C Programming

- Basic Concepts
  - **Basic Output**
    - Some output examples



`printf(“ABC”);`	ABC (cursor after the C)
`printf(“%d\n”,5);`	5 (cursor at start of next line)
`printf(“%c %c %c”,’A’,’B’,’C’);`	`A B C`

`printf(“From sea ”);`
`printf(“to shining “);`
`printf (“C”);`
	`From sea to shining C`

`printf(“From sea \n”);`
`printf(“to shining \n“);`
`printf (“C”);`
	`From sea`
	`to shining`
	`C`

`leg1=200.3; leg2=357.4;`
`printf(“It was %f`
`miles”,leg1+leg2);`
	`It was 557.700012 miles`

`num1=10; num2=33;`
`printf(“%d\t%d\n”,num1,num2);`
	`10      33`

Basics of C Programming

- **Tasks**
  - Setup the C programming environment as explained below
  - Finish the provided assignments that will be provided
  - Prepare for Test One

Environment Setup

- Set up the environment that will allow you to write a C program.
  - You have to install two software in your computer:
    - Text Editor
    - C Compiler.
  - Text Editor - Is a software used to type a program.
    - Examples of editors include
      - Windows Notepad
      - Notepad++
      - gedit
      - vi

      etc.

Environment Setup

- The files you create with your editor are called the source files
- They contain the program source codes
- The source files for C programs are named with the extension ".c"

Environment Setup

- You may use IDEs available

  (IDE - Integrated Development Environment)

  - An IDE is a software that combines basic tools required to write and test software
  - It has built-in functions like debugging, code completion, compiling and syntax highlighting
  - The main use of IDE is to provide different components of software applications while developing the program
  - Examples: ***CodeBlocks, Eclipse, NetBeans***