To:
Tenouk

C LAB WORKSHEET 5_1
C/C++ Variable, Operator And Type 2

Items in this page:

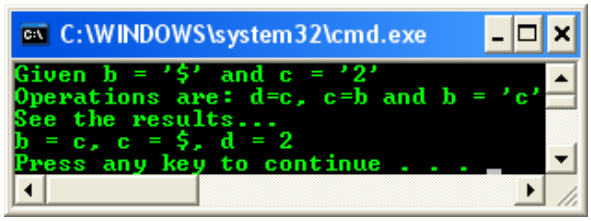
1. In this page more variable, data type and operator program examples, questions and activities.
2. Character and string data types.
3. Tutorial references are: C/C++ intro & brief history, C/C++ data type 1, C/C++ data type 2, C/C++ data type 3 and C/C++ statement, expression & operator 1, C/C++ statement, expression & operator 2 and C/C++ statement, expression & operator 2.

- Character is another basic data type and strings are combinations of characters. Show the output for the following program.

```
// needed for printf()
#include <stdio.h>

int main()
{
    // declare variables and initialize some of them
    // some compiler ask you to initialize all the variables...
    // in this case just initialize them with dummy value..
    // for example: int x = 0, float y = 0.00 etc.
    char b = '$', c = '2', d;
    printf("Given b = \'$\' and c = \'2\'\n");
    printf("Operations are: d=c, c=b and b = \'c\'\n");
    d = c;      // Statement #1
    c = b;      // Statement #2
    b = 'c';    // Statement #3
    printf("See the results...\n");
    printf("b = %c, c = %c, d = %c\n", b, c , d);
    return 0;
}
```

A sample output:



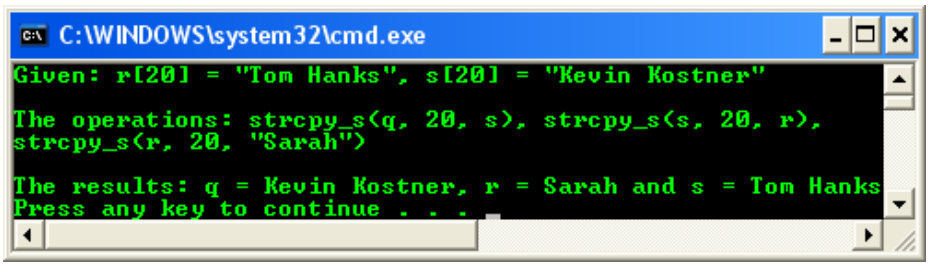
- In statement 3 the character 'c' is being assigned and not the variable c, as it was in statement 1. Also note that in statement 1, d is being assigned the character 2 and not the integer 2.
- To assign the string s to the string t (copy a string), one may use strcpy_s(t, dest. size, s) function family and the equal sign as before. Show the output of the following program.

```
// needed for printf()
#include <stdio.h>
// needed for strcpy_s() and their family
#include <string.h>

int main()
{
    // declare variables and initialize some of them
```

```
    // some compiler ask you to initialize all the variables...
    // in this case just initialize them with dummy value..
    // for example: int x = 0, float y = 0.00 etc.
    char q[20], r[20] = "Tom Hanks", s[20] = "Kevin Kostner";
    printf("Given: r[20] = \"Tom Hanks\", s[20] = \"Kevin Kostner\"\n");
    printf("\nThe operations: strcpy_s(q, 20, s), strcpy_s(s, 20, r), \n"
        "strcpy_s(r, 20, \"Sarah\")\n");
    // the destination string must be large enough to hold the
    // source string, including the terminating null character.
    // if the non-secure version is used, such as strcpy()
    // compiler may generate warning...
    strcpy_s(q, 20, s);
    strcpy_s(s, 20, r);
    strcpy_s(r, 20, "Sarah");
    printf("\nThe results: q = %s, r = %s and s = %s\n", q, r, s);
    return 0;
}
```

A sample output:



- Firstly, “Kevin Kostner” is assigned to q, “Tom Hanks” is assigned to s, and then “Sarah” is assigned to r.
- Characters have single quotes, strings have double quotes, integers are whole numbers and float contains decimal points. The following statements just numbers but in C/C++ programming they have different meanings. Label the following data types appropriately.

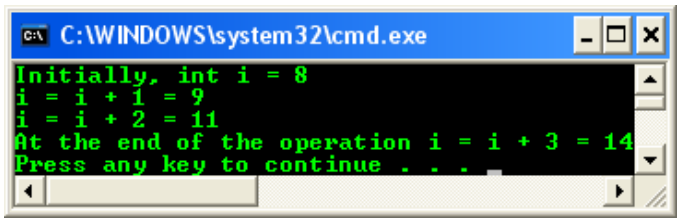
3.1234 = _____ **Ans:** float
"3214.55" = _____ **Ans:** string
'5' = _____ **Ans:** char
1243 = _____ **Ans:** int

- Show the output of the following program. See the changes of the variable values during the program execution that is why we call it a **variable**.

```
// needed for printf()
#include <stdio.h>
// needed for strcpy_s() and their family
#include <string.h>

int main()
{
    // declare variables and initialize some of them
    // some compiler ask you to initialize all the variables...
    // in this case just initialize them with dummy value..
    // for example: int x = 0, float y = 0.00 etc.
    int i = 8;
    printf("Initially, int i = %d\n", i);
    i = i + 1;
    printf("i = i + 1 = %d\n", i);
    i = i + 2;
    printf("i = i + 2 = %d\n", i);
    i = i + 3;
    printf("At the end of the operation i = i + 3 = %d\n", i);
    return 0;
}
```

A sample output:



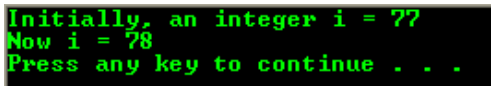
Questions And Activities:

- Run the following exercises, record the outputs, answer the questions and try to understand why the outcome is like that.

1. The data type for the variable i is an integer or a whole number.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 77;
    printf("Initially, an integer i = %d
\n", i);
    i = i + 1;
    printf("Now i = %d\n", i);
    return 0;
}
```



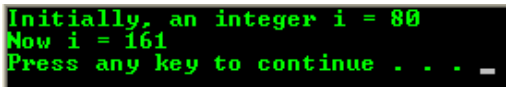
   a. What does the statement i = i + 1; do to the value of i?

   a. The i = i + 1 add 1 to the current value of integer i (77).

2. Show the following program output. The lines with the equal signs (=) are called **assignment statements**.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 77;
    i = i + 3;
    printf("Initially, an integer i = %d
\n", i);
    i = i + i + 1;
    printf("Now i = %d\n", i);
    return 0;
}
```



   a. Why is i printed 80 instead of 77 in the first printf()?
   b. What is the value of i while the last printf() is being called? Why?

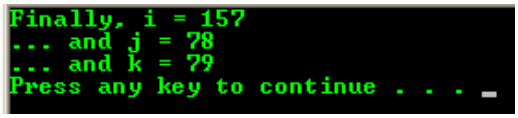
   a. Because of the statement i = i + 3; Then 77 + 3 = 80. So the current value of i is 80.
   b. It is 161. it is because of the statement i = i + i + 1; The current value of variable i before this statement been executed is 80, then 80 + 80 + 1 = 161.

3. Show the output and answer the following questions.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 77, j, k;
    j = i + 1;
    k = j + 1;
    i = k + j;
    printf("Finally, i = %d\n", i);
    printf("... and j = %d\n", j);
    printf("... and k = %d\n", k);
    return 0;
}
```

a. What is the final value of j and how did it obtain that value?
b. What is the final value of k and how did it obtain that value?
c. What is the final value of i and how did it obtain that value?



a. 78. From the assignment of j = i + 1; statement and the current value of i is 77 then j = 77 + 1 = 78.
b. 79. It is obvious that k = j + 1 and the current value of j is 78, so k = 78 + 1 = 79.
c. 157. The assignment statement for i is i = k + j. The current value of k is 79 and j is 78 so i = 79 + 78 = 157.

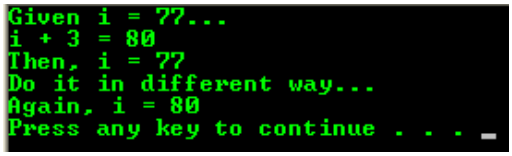
In this example we can see that the value of variable change during the program execution. It depend on the operation(s) done on the variable and the order of the statement in the program. C code executed sequentially, line by line. What we concern here is the current value of the variable before the statement been executed.

4. Try the following codes and answer the questions.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 77;
    printf("Given i = 77...\n");
    printf("i + 3 = %d\n", i + 3);
    printf("Then, i = %d\n", i);
    printf("Do it in different way...
\n");
    i = i + 3;
    printf("Again, i = %d\n", i);
    return 0;
}
```

a. In which statement is i declared as an integer?
b. What is the initial value of i?
c. Did the first printf() assign a new value to i?
d. Did i = i + 3; assign a new value to i?



a. int i = 77;
b. 77
c. No. This just a literal string printed out to the standard output by printf().
d. Yes. This is an assignment statement. Add 3 to the current value of i and assigned the new value to variable i.

5. Run the following program and answer the questions.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 12;
    printf("Initially i = %d\n", i);
    printf("Then, i * 3 - 4 = %d\n", i
* 3 - 4);
    printf("Different answer? i * (3 -
4) = %d\n", i * (3 - 4));
    printf("Another one, i / 2 + 4 * 2
= %d\n", i / 2 + 4 * 2);
    printf("Similar expression but
different answer? i / (2 + 4) * 2 = %
d\n", i / (2 + 4) * 2);
    printf("Finally, i = %d\n", i);
    return 0;
}
```

a. After i was initialized to 12, was it value ever changed?
b. What is the symbol used to multiply? To divide?
c. In the second printf(), which operation, multiply or subtract was done first?
d. In the third printf(), which was done first, multiply or subtract? Why?
e. In fourth printf(), which operation was done last?
f. Which of the following has the highest, next highest and the lowest precedence/ priority? That is which one will be done first, next and last?

    * and /; + and -; ()

The priority execution of the expression in C/C++ depends on the operator precedence and is done automatically by compiler if the parentheses/brackets (()) are not used. In order to make your code readable or to produce a correct output, it is better to use parentheses/brackets that having the highest priority. So, forget about operator precedence just use brackets.

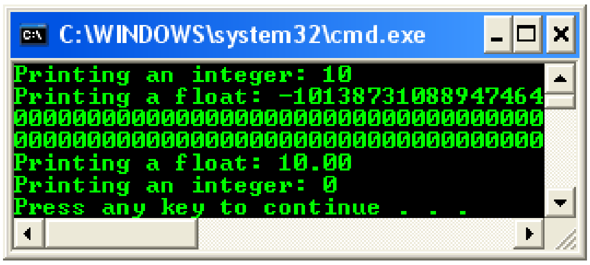
```
Initially i = 12
Then, i * 3 - 4 = 32
Different answer? i * (3 - 4) = -12
Another one, i / 2 + 4 * 2 = 14
Similar expression but different answer? i / (2 + 4) * 2 = 4
Finally, i = 12
Press any key to continue . . .
```

a. Yes.
b. Multiply - * (asterisk) and divide - / (forward slash)
c. Multiply was done first. This is a operator precedence. Multiply is higher precedence than subtract operator.
d. Subtract first because of the bracket. Bracket is higher than mathematical operators.
e. Multiplication followed by division and then addition.
f. *, /, + and - (from highest to lowest.

6. Playing with floating point numbers. Try the following code and answer the questions.

```c
// needed for printf()
#include <stdio.h>

int main()
{
      printf("Printing an integer: %d\n", 10);
      printf("Printing a float: %.2f\n", 10);
      printf("Printing a float: %.2f\n", 10.00);
      printf("Printing an integer: %d\n", 10.00);
      return 0;
}
```



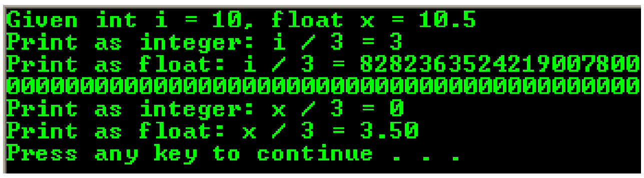
a. Which line of codes that print correct values? Keep in mind that your compiler doesn't generate any warning or error but some of the output is rubbish or zero if format specifier is not used properly.
b. Do you use a %d or a %.2f to print an integer?
c. Do you use a %d or a %.2f to print a float?

a. printf("Printing an integer: %d\n", 10); and printf("Printing a float: %.2f\n", 10.00);
b. %d.
c. %.2f.

7. More floating point program. Run and answer the questions.

```c
// needed for printf()
#include <stdio.h>

int main()
{
      int i = 10;
      float x = 10.5;
      printf("Given int i = 10, float x = 10.5\n");
      printf("Print as integer: i / 3 = %d\n", i / 3);
      printf("Print as float: i / 3 = %.2f\n", i / 3);
      printf("Print as integer: x / 3 = %d\n", x / 3);
      printf("Print as float: x / 3 = %.2f\n", x / 3);
      return 0;
}
```



a. To print an integer result, what format specifier must be used?
b. To print a floating point result, what format specifier must be used?

a. %d.
b. %f.

8. Remember that i is defined to be an integer or a whole number. It cannot store values with decimals (for fractions). Notice in these printf()'s that some expressions are evaluated as integers and some as floats. Also, some format specifiers are integers and some are floats.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i;
    i = 10;
    printf("Given int i = 10\n");
    printf("Print as int: i / 2 = %d\n",
i / 2);
    printf("Print as int: i / 3 = %d\n",
i / 3);
    printf("Print as float: i / 3 = %.2f
\n", i / 3);
    printf("Print as float: i / 3.0 =
%.2f\n", i / 3.0);
    printf("Print as float: (i + 0.0) / 3
= %.2f\n", (i + 0.0) / 3);
    printf("Print as int: i * 3 = %d\n",
i * 3);
    printf("Print as float: i * 3.0 =
%.2f\n", i * 3.0);
    return 0;
}
```

```
Given int i = 10
Print as int: i / 2 = 5
Print as int: i / 3 = 3
Print as float: i / 3 = -0.00
Print as float: i / 3.0 = 3.33
Print as float: (i + 0.0) / 3 = 3.33
Print as int: i * 3 = 30
Print as float: i * 3.0 = 30.00
Press any key to continue . . .
```

a. In C/C++, why do you think 10 divided by 3 is equal to 3 and not 3.33?
b. i/3 evaluates to an integer or a float?
c. i/3.0 evaluates to an integer or a float?
d. How do you think (i * 1.0)/3 will be evaluated?
e. To see the fractional part after dividing two items, what must be true of one of those items?

a. Because a format specifier %d, that is for integer was used instead of % f for float and integers numbers (for numerator and denominator) were used instead of floats in the division.
b. an integer.
c. a float.
d. Evaluated as a float because 1.0 was used.
e. One of the items must be a float.

9. Try the following program example.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 10;
    float x = 10.5; // a dummy initial
value
    x = i / 3;  // warning, int to float
    printf("Given int i = 10, float x =
10.5\n");
    printf("Print as float: i / 3 = %.2f
\n", x);
    x = i / 3.0;      // warning, double
to float
    printf("Print as float: i / 3.0 =
%.2f\n", x);
    return 0;
```

```
Given int i = 10, float x = 10.5
Print as float: i / 3 = 3.00
Print as float: i / 3.0 = 3.33
Press any key to continue . . .
```

a. A floating point value because the result for the division was set to floating point. However the value has been truncated because items (numerator and denominator) used in the division are integers. Compiler will do an automatic conversion to float for float and/or integer

```
}
```

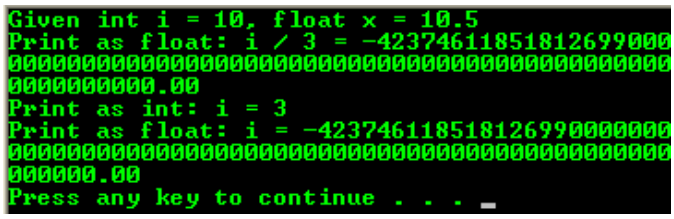
a. In x = i/3, was the evaluation of i/3 an integer value or a floating point value?
b. Why was the result from the second printf () more accurate than the result from the first printf()?

items.
b. Because one of the item (denominator) used in the division is a float.

10. Let try the following program.

```
// needed for printf()
#include <stdio.h>

int main()
{
    int i = 10;
    float x = 10.5;
    i = x / 3;  // warning
    printf("Given int i = 10, float x = 10.5\n");
    printf("Print as float: i / 3 = %.2f \n", i); // rubbish
    printf("Print as int: i = %d\n", i);
    i = x / 3.0;      // warning
    printf("Print as float: i = %.2f\n", i); // rubbish
    return 0;
}
```



a. In i = x/3, was the evaluation of x/3 an integer value or a floating point value?
b. Can we store a floating point value in an integer variable?
c. In an arithmetic expression, such as i/2 + 1, if all the items are integers, then the result is of what type?
d. If at least one item is a float, then the result is of what type?
e. When dividing two integers, the result will contain a fractional portion only if it is assigned to a floating point variable. True or False?
f. When dividing two items and one is a float, the result will be an integer. True or False?

a. The output is rubbish.
b. No.
c. Integer.
d. Float.
e. True.
f. False.