To:
Tenouk

# C LAB WORKSHEET 5_2
# C/C++ Variable, Operator And Type 3

Items in this page:



11. Now, let try characters and strings. A string is nothing more than a collection of characters. The number of characters are in the brackets, [ ] minus one (reserved for terminating null, or **null character** or '\0' to mark the end of the string) gives the length of the string. The following t, an array variable can store a string that is 10 characters long.

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

int main()
{
    char a = 'W';
    char t[11] = "PeAcE";
    printf(" a = %c\n", a);
    printf(" t = %s\n", t);
    return 0;
}
```

```
 a = W
 t = PeAcE
Press any key to continue . . .
```

a. Name the character variable.
b. Name the string variable.
c. Is a character enclosed between single quotes or double quotes?
d. Is a string enclosed between single quotes or double quotes?

a. a.
b. t.
c. single quotes.
d. double quotes.

12. Since strings are collection of characters, not single items, they need to be treated differently from other basic data type such as integers, floats and characters. Let try strcpy_s() function (this is a secure version of strcpy()), which defined in the string.h header file.

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

int main()
{
    char a = 'W', b;
    char t[20] = "The copied PeAcE", q[20];
    b = a;
    strcpy_s(q, 20, t);
    printf("b = %c\n", b);
    printf("t = %s\n", t);
    return 0;
}
```

```
b = W
t = The copied PeAcE
Press any key to continue . . .
```

a. How is a character assigned to a character variable?
b. How is a string assigned to a string variable?

a. A character must be in single quotes.
b. A string must be in double quotes.

13. Let swap the values of t and r.

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

int main()
{
    char t[20] = "The copied PeAcE", q[20], r[20];
    strcpy_s(r, 20, "Copied to r");
    strcpy_s(q, 20, t);      // t to q
    strcpy_s(r, 20, q);     // q to r
    strcpy_s(t, 20, r);      // r to t
    printf("t = %s\n", t);
    printf("q = %s\n", q);
    printf("r = %s\n", r);
```

```
t = The copied PeAcE
q = The copied PeAcE
r = The copied PeAcE
Press any key to continue . . .
```

a. "Copied to r" string.
b. "The copied PeAcE" string.
c. "The copied PeAcE" string.
d. "The copied PeAcE" string.

In this program, the t array contains "The copied PeAcE" string. Then "Copied to r" string been copied to array r. Next, array t been copied to

```
    return 0;
}
```

a. What value was assigned to r?
b. What value was assigned to q?
c. What value was assigned to t?
d. Now what value was assigned to r?

array q, so q contains "The copied PeAcE". Next, array q been copied to r so we have "The copied PeAcE" string in r. Finally r been copied to t so we have "The copied PeAcE" string in r. At the end all the array will contain "The copied PeAcE" string.

14. Show the output for the following code. Study the source code and the output.

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

void main(void)
{
    int i = 3, j = 5;
    i = i + 1;
    j = j + i + 1;
    printf("%d %d\n", i + 1, j - 1);
    printf("%d %d\n", i, j);
}
```

```
5 9
4 10
Press any key to continue . . .
```

For the first printf() the current value of i is 4 and j is 10 as shown in the second printf().

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

void main(void)
{
    double x = 13.5, y = -23.2;
    x = (x / 2) * -1 + y;
    y = 13 / 2 - 2;
    printf("%f %.2f\n", x, y);
}
```

```
-29.950000 4.00
Press any key to continue . . .
```

Considering the operator precedence, i n the first expression we have:
x = (13.5/2)*-1+(-23.2)
x=6.75*-1+(-23.2)
x=-6.75-23.2
x=-29.950000 (default to 6 floating point precision)

In the second expression:
y=13/2 -2
y=6-2 (truncated to an integer because both numerator and denominator are integers)
y=4.00 (to 2 floating point precision)

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

void main(void)
{
    // Note that we just initialize b to a space, s to a comma just for dummy...
    char a = 'b', b = ' ', s[21] = ",", t[21] = "OvEr", r[21] = "RoLl";
    printf("a = %c b = %c r = %s s = %s t = %s\n", a, b, r, s, t);
    b = a;
    a = 'c';
    strcpy_s(s, 21, r);
    strcpy_s(r, 21, t);
    strcpy_s(t, 21, s);
    printf("a = %c b = %c r = %s s = %s t = %s\n", a, b, r, s, t);
}
```

```
a = b b =   r = RoLl s = , t = OvEr
a = c b = b r = OvEr s = RoLl t = RoLl
Press any key to continue . . .
```

Before the first strcpy_s(), the current value for the variables are:
a='c', b='b', s=,, t="OvEr" and r="RoLl"
The first strcpy_s() copies "RoLl" string to s so s contains "RoLl" string. The second strcpy_s() copies "OvEr" string to r so r contains "OvEr" string and the third strcpy_s() copies "RoLl" string to t so t contains "RoLl" string.

15. Declare me as an integer. Assign it the value of 40. Declare you as an integer and assign it the value of 20. Declare average as a float and assign it the average of me and you. Then using one printf() for each line, print the following. The numbers are printed using the values of the variables. The last line starts on a tab position.

```
----------Output----------
I am 40.
You are 20
    We are around 30!
```

```
#include <stdio.h>

void main(void)
{
    int me = 40, you = 20;
    float average = 0.0;
    // the (float) is a simple C type promotion. Here we promote
    // the integer type to float before assign it to
    // average variable
    average = (float)((me+you)/2);
    printf("I am %d\n", me);
    printf("You are %d\n", you);
    printf("\tWe are around %.0f\n", average);
}
```

```
I am 40
You are 20
        We are around 30
Press any key to continue . . .
```

16. Declare a string called name that can hold up to 20 characters including the terminating null character. Declare rate as a float and hours as an integer. Now assign “Garfield”, 10.0 and 20 to these variables respectively. Without creating any new variables, print the following using the values of name, rate and hours. Use one printf() for each line that is printed.

```
---------Output----------
His name is Garfield,
Rate is 10.00 and the hours worked was 20
Garfield made 200.00 dollars.
```

```
#include <stdio.h>
void main(void)
{
   char name[20]= "Garfield";
   float rate = 0.0;
   int hours = 0;
   rate = 10.0;
   hours = 20;
   printf("His name is %s\n", name);
   printf("Rate is %.2f and the hours worked was %d\n", rate, hours);
   printf("Garfield made %.2f dollars.\n", rate*hours);
}
```

```
His name is Garfield
Rate is 10.00 and the hours worked was 20
Garfield made 200.00 dollars.
Press any key to continue . . .
```

17. Study the first program example in this Module. Find the codes that you don’t understand and discuss with your partner/group member to find the answers. Then create four questions and answers.

18. From time to time we will use the functions from the standard library that normally come together with your compiler. For example, we can use the mathematical functions such as square root (sqrt()) and power-to (pow()) functions from math.h header. Run the following program and show the output.

```
// needed for printf()
#include <stdio.h>
// needed for math functions such as sqrt()
#include <math.h>

void main(void)
{
    // study this funny part, add 1 before and add 1 after...
    int i = 1, j;
    ++i; // i = i + 1
    printf("i = %d\n", i);
    j = ++i; // j = i = i + 1 - i already equal to 2
    printf("j = %d\n", j);
    j = i++; // assign i to j and then i = i + 1
    printf("i = %d j = %d\n", i, j);
    // using functions from standard library, math.h
    printf("Square root of 9 is %f\n", sqrt(9));
    printf("2 raised to the power of 3 is %f\n", pow(2, 3));
}
```

```
i = 2
j = 3
i = 4 j = 3
Square root of 9 is 3.000000
2 raised to the power of 3 is 8.000000
Press any key to continue . . .
```

---

19. By considering the operator precedence, convert the following equation to C expression.

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

```
((3 + (4*x)) / 5) - (10*((y-5)*(a+b+c)))/x + (9*((4/x) + ((9+x)/y)))
```

$$\sqrt{x^2 + y^2}$$

```
sqrt(pow(x,2) + pow(y,2))
```

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
(-b + sqrt((b * b) - (4 * a * c))) / (2 * a)
```

20. Convert the following C expression to mathematical expression.

```
sqrt(pow(x,2) + pow(y,2))
```

$$\sqrt{x^2 + y^2}$$

21. Based on the C operator precedence and associativity, state the execution order of the following expressions.

a + b + c + d + e

a + b * c - d / e

a / (b + c) + d % e

a / (b * (c + (d - e)))

a + b + c + d + e
1 2 3 4

a + b * c - d / e
3 1 4 2

a / (b + c) + d % e
2 1 4 3

a / (b * (c + (d - e)))
4 3 2 1

22. Convert the following mathematical expressions to C expressions.

a. $rate^2 + delta$

b. $2(salary + bonus)$

c. $\frac{1}{time + 3mass}$

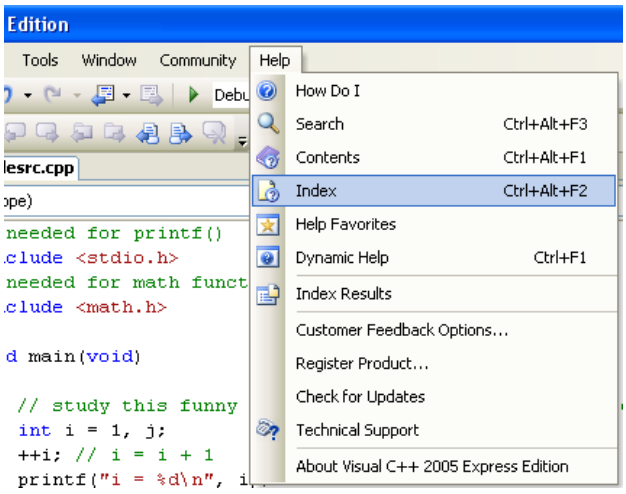
d. $\frac{a - 7}{t + 9v}$

a. (rate*rate) + delta
b. 2*(salary + bonus)
c. 1/(time + (3*mass))
d. (a - 7) / (t + (9 * v))

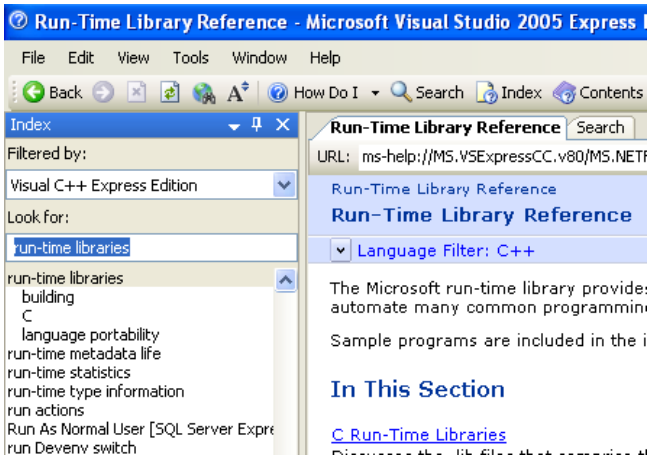
**Using Compiler's Documentation**

The best resource of programming is the documentation that comes together with your compiler. Using the compiler's documentation means we refer to the authoritative source of reference. In practice #18 you have been introduced with sqrt() and pow() functions that defined in math.h header file. From the previous lab practice you should already know how to find the math.h header file in your machine.
What about the information for sqrt() and pow() functions? How to use those functions? How to write a proper syntax? Now you will learn how to dig that information.

1. While in your Visual C++ 2005 EE, select **Help** → **Index** menu (make sure you have installed the MSDN library).

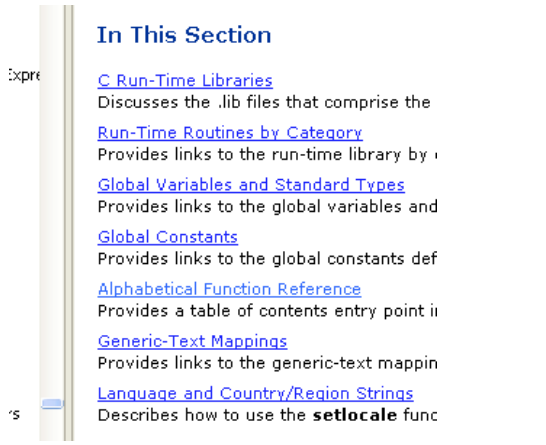



2. Type "**run-time libraries**" in the **Look for:** field and then click the highlighted "**run-time libraries**" just under the **Look for:** field. The information page for the run-time libraries is displayed on the right window.

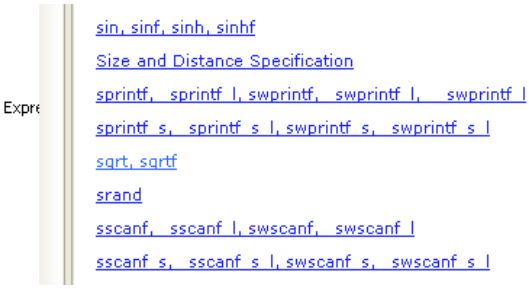



3. You can see that for C programming, Visual C++ 2005 (Microsoft) called **C Run-Time** (**CRT**) Libraries. The Microsoft CRT contains much more than the Standard C Library. There is a lot of information that you need to filter out to find your specific information. For this practice we just want to find information for square root function (sqrt()).

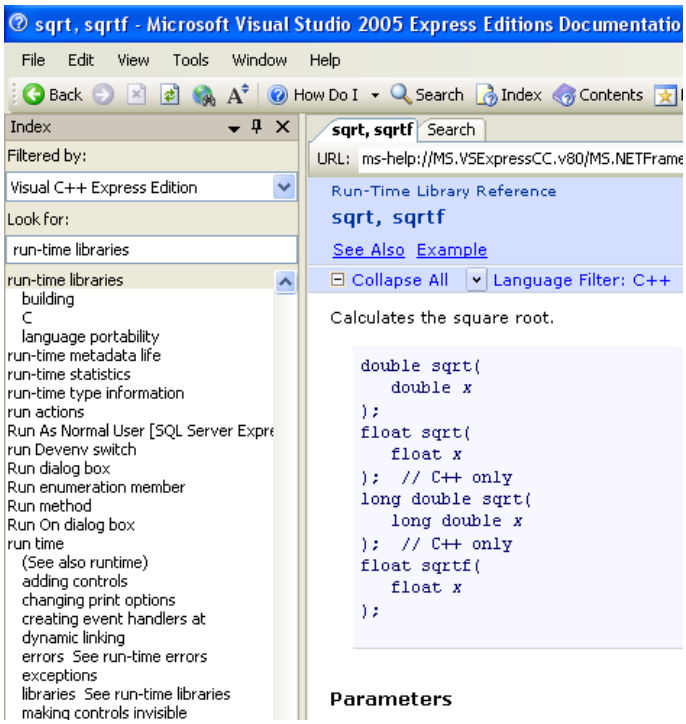
4. Click the **Alphabetical Function Reference** link.




5. Next, scroll down the page until you find sqrt, sqrtf and click the link.




6. You will find a complete sqrt() information that includes: the prototype, parameter, return value, required header file, program example and .Net framework equivalent class.




**Question:** Fill up the following tables for sqrt(), pow() and etc. functions. This will become your simplified reference.

| Item | Description |
|---|---|
| Function | sqrt(). |
| Use | To calculate the square root. |
| Prototype | double sqrt( double x ); |
| Parameters | x - Nonnegative floating-point value. |
| Example | double num = 45.35, answer;<br>answer = sqrt( num ); |
| Return value | The sqrt function returns the square-root of x. If x is negative, sqrt returns an indefinite, by default. |
| Include file | <math.h> |
| Remark | - |

| Item | Description |
|---|---|
| Function | pow(). |
| Use | To calculate x raised to the power of y. |
| Prototype | double pow( double x, double y ); |
| Parameters | x - Base.<br>y - Exponent. |
| Example | double x = 2.0, y = 3.0, z;<br>z = pow( x, y ); |
| Return value | Returns the value of xy. No error message is printed on overflow or underflow. |
| Include file | <math.h> |
| Remark | - |

| Item | Description |
|---|---|
| Function | strcpy (). |
| Use | To copy a string. These functions are deprecated because more secure versions are available: strcpy_s(), wcscpy_s(), _mbscpy_s(). |
| Prototype | char *strcpy( char *strDestination, const char *strSource ); |
| Parameters | strDestination - Destination string.<br>strSource - Null-terminated source string. |
| Example | char string[80];<br>strcpy( string, "Hello world from " ); |
| Return value | Returns the destination string. No return value is reserved to indicate an error. |
| Include file | <string.h> |
| Remark | The strcpy() function copies strSource, including the terminating null character, to the location specified by strDestination. The behavior of strcpy is undefined if the source and destination strings overlap.<br>Because strcpy() does not check for sufficient space in strDestination before copying strSource, it is a potential cause of buffer overruns. Consider using strcpy_s() instead.<br>wcscpy() and _mbscpy() are wide-character and multibyte-character versions of strcpy () respectively. The arguments and return value of wcscpy() are wide-character strings; those of _mbscpy() are multibyte-character strings. These three functions behave identically otherwise.<br>In C++, these functions have template overloads that invoke the newer, secure counterparts of these functions. |

| Item | Description |
|---|---|
| Function | strcpy_s() |
| Use | Copy a string. These are versions of strcpy(), wcscpy(), _mbscpy() with security enhancements. |
| Prototype | errno_t strcpy_s( char *strDestination, size_t sizeInBytes, const char *strSource ); |
| Parameters | strDestination - Location of destination string buffer<br>sizeInBytes, sizeInWords - Size of the destination string buffer.<br>strSource - Null-terminated source string buffer. |
| Example | char string[80];<br>strcpy_s( string, "Hello world from " ); |
| Return value | Zero if successful; an error otherwise. |
| Include file | <string.h> |
| Remark | The strcpy_s() function copies the contents in the address of strSource, including the terminating null character, to the location specified by strDestination. The destination string must be large enough to hold the source string, including the terminating null character.<br>The behavior of strcpy_s() is undefined if the source and destination strings overlap.<br>wcscpy_s() and _mbscpy_s() are wide-character and multibyte-character versions of strcpy_s () respectively. The arguments and return value of wcscpy_s() are wide character strings; those of _mbscpy_s() are multibyte character strings. These three functions behave identically otherwise.<br>If strDestination or strSource is a null pointer, or if the destination string is too small, the invalid parameter handler is. If execution is allowed to continue, these functions return EINVAL and set errno to EINVAL. Upon successful execution, the destination string will always be null terminated.<br>In C++, using these functions is simplified by template overloads; the overloads can infer buffer length automatically (eliminating the need to specify a size argument) and they can automatically replace older, non-secure functions with their newer, secure counterparts. |

- Unfortunately, in this edition of Visual C++ 2005 EE, Tenouk cannot find the documentation for the header files that available in Standard C Library. Well, you can find it in the internet domain: C Standard Library. It is standardized in 1999 that is why sometime and somewhere you will find it is referred as C99. Remember that, other than the Standard version, it is implementation specific. This means if you use Microsoft extension of C, you can only compile the code using Microsoft compiler.
- Keep in mind that Microsoft C Run-Time contains a mixed of the Standard (ANSI/ISO/IEC – ISO/

IEC 9899:1999) and non-standard (Microsoft implementation) C library. If you already familiar with Standard C library, from the list of the functions reference, you should already noticed that for Microsoft implementation, the function name is preceded by an underscore ( _ ) or double underscore ( __ ). The standard one still retains the similar name and functionalities but it is Microsoft version. For the standard libraries, you can find and use it in any C/C++ compilers.

- For Standard C++ Library documentation, type “Standard C++ Library” in the **Look for:** field of the VC ++ EE Help. This C++ Standard Library used by Microsoft is based on the ANSI/ISO/IEC (ISO/ IEC 14882:2003).
- For Windows’s Graphic User Interface (GUI) information, you can try searching the “**MFC libraries**” in the documentation. Microsoft Foundation Classes is totally Microsoft implementation though it is based on C++. Unfortunately it is not available in Tenouk’s copy of the Visual C++ 2005 EE. You can try finding it in the Visual Studio 2005 Express Edition.