ARRAY & FUNCTIONS

Assuming that you have already had the basics in:

- **simple control structure including nested loops**
- **Basic understanding of primitive data types.**
- **Know about what ASCII table**.
- **Know how to write functions**

MUST read through the "Review" section before doing the exercises.

CONTENTS

REVIEW ON ARRAY

Take the construction like this:

```
<data type> variable[ number  of elements ]
```

store homogeneous elements at contiguous locations

Simple example:

Simple declaration and initialization for 1-D array

```
int arr[5];       // data size of "int" * 5
```

```
for (i=0; i< 5; i++)
   arr[i] = i*2;
```

Bad habit to hard code the "5". Use "sizeof(…)

```
sizeof(int) == 4
sizeof (arr) = 4*5
# of elements = sizeof(arr) / sizeof(arr[0])
```

```
for (i=0; i< sizeof(arr) / sizeof(arr[0]); i++)
   arr[i] =  <something>;
```

More samples:

```
char arr[ ] = {'c','o','d','e','\0'};
char arr[ ] = "code";
int arr[100] = 0;
int arr[100] = 1;    // hmm… watch out . this does not set every element to 1, every byte to 1 instead.
```

Pay attention to the boundary

e.g. int arr[4];

valid : arr[3] = 5;

invalid : arr[4] = 5;

Sample code

```
void main()
{
        float mass[5] = { 10, 14, 20, 50, -10 };
        float sum = 0.0;
        int i;
        cout << "average of ";
        for (i = 0; i < 5; i++)
        {
                cout << mass[i] << ", ";
                sum = sum + mass[i];
        }
        cout << " = " << sum / i << endl;
}
```

Simple declaration and initialization for 2D array

```
int arr[2][3];       // data size of arr = 2x3x4

valid:  int arr[ 2 ][ 3 ] = { { 0, 0, 1 }, { 0, 0, 2 } };

valid:  int arr[   ][ 3 ] = { { 0, 0, 1 }, { 0, 0, 2 } };

invalid:  int arr[ 2 ][  ] = { { 0, 0, 1 }, { 0, 0, 2 } };
```

ENUMERATION TYPE

Macros method:	Use Enumeration instead:
#define NORTH 0 #define SOUTH 1 #define EAST 2 #define WEST 3	enum Directions { NORTH, EAST, SOUTH, WEST}; Directions dir;
e.g. Valid : int dir = NORTH; dir = WEST ; dir = (Directions)3;	e.g. Valid : dir = NORTH; dir = WEST ; dir = (Directions)3;
Also Valid: dir = (Directions) 4;	WRONG! : dir = (Directions) 4;

MAY USE IT FOR INDEXING ARRAY

e.g. :

```
char directions[WEST];
sizeof(directions) is 3

char directions[WEST+1];
sizeof(directions) is 4

char directions[ ][10] = { "NORTH", "EAST", "SOUTH", "WEST" };
sizeof(directions) == 40
```

DEMONSTRATE FURTHER USAGE:

Sample 1:

```
enum enDir { N, E, S, W,  Last};
int sDir[ Last ];        // sizeof( sDir ) ==16
```

Sample 2:

```
enum enDir { N, NE, E, SE, S, SW, W, NW, Last};
int sDir[ Last ];        // sizeof( sDir ) ==32
```

Sample 3: (add onto sample 2)

```
char sDir[ Last ][10] = { "N", "NE", "E", "SE", "S", "SW", "W", "NW" };
```

```
;       // sizeof(  sDir ) == 80

enDir d;

for ( d = N;    d < Last;     d = (enDir)(d+1) )                // ie. From 0 to 7
{
            if ( checkDir(dir, d) )
                  printf("%s  is set\n", sDir[d]);
            else
                  printf("%s  is not set\n", sDir[d]);
}
```

Remark: so, you can add any # of elements before the "Last", your loop will always work without overflow.

FUNCTIONS

Take the construction like this:

```
<data type> function_name(list of parameters){  expressions }
```

Simple example:

Example 1	void doWork()
	{
	  ….
	  return; // does not return any value
	}

Example 2	float sum(float data[ ] , int num)
	{ int i;
	  float total=0.0;
	  for (i=0 ; i<num; i++){
	  total = total + data[i];
	  }
	  return total;
	}

Example 3	float calArea(float a, float b, char kind)
	{
	  float area;
	  switch (kind):
	  {
	  case 's' : area = a * b;
	  break;
	  case 't' : area = a * b / 2.0;
	  break;
	  …
	  }
	  return area;
	}

Example 4	Use the Object struct sample above:
	
	float calForce(Object ob)
	{
	  return (ob.velocity/ ob.time)* ob.mass;
	}
	
	void main()
	{
	  Object bots[2] = { { 50, 9.5, 10.5 }, { 20, 20, 5.5 } };
	  for(int i = 0; i< 2; i++)
	  cout << "object-" << i << ": " << calForce(bots[i[) << endl;
	}

Example 5	typedef struct Child {
	char name[7];
	char gender;
	int age;
	void setChild(int a, char s[], char g)
	{
	  strcpy(name, s, sizeof(name));
	  gender = g;
	  age = a;
	}
	} Child;
	
	void main()
	{
	  Child ct;
	  ct.setChild(10, 'f', 20);
	  printf ("%d\n", ct.gender);
	
	}

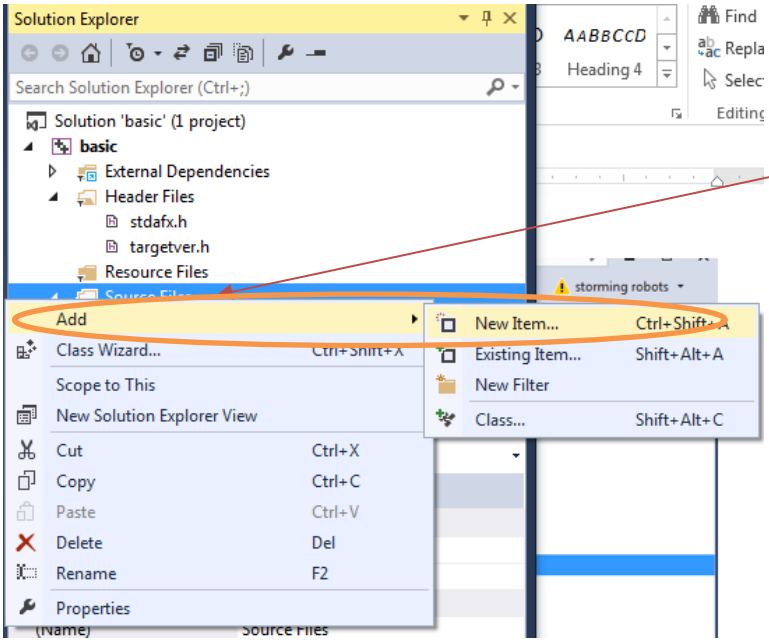
BUILDING A PROJECT WITH MULTIPLE FILES AND HEADERS

How to add header and new programs into your project:

Using Eclipse:

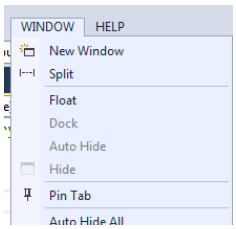
You just need to create *.cpp and *.h in the same project folder.

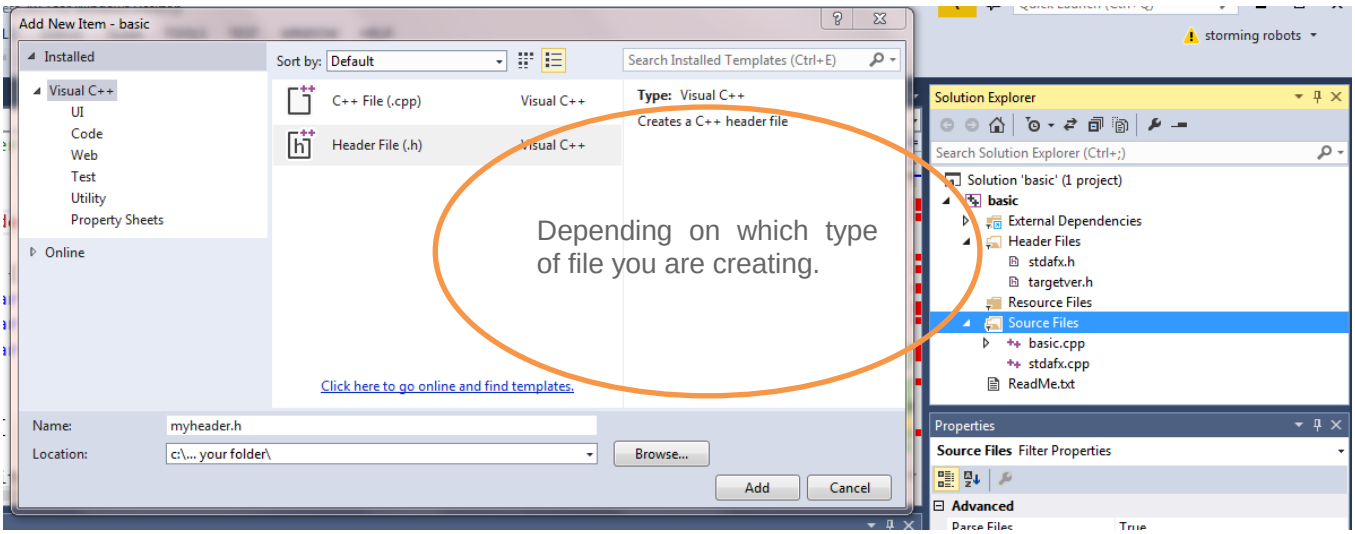
Using Visual Studio:



Source Files

If you do not see this Solution Explorer, do :





EXERCISES

1) Reverse the value stored in an array of x[…].
   Reverse the list of value; e.g.

   X[...] contains 10,4,1, 9. After the function reverse(…)
   X[...] contains 9,1,4,10

2) Define an array char a[26] and fill this array with values 'A', 'B', 'C', to 'Z' in a simple loop with only one expression. This expression should not contain any numeric constant.

   e.g.

   for (i=0; i<sizeof(a); i++)
   the expression goes here….

   Yes. Only one!

   Sample output:

   A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z

   Tip: remember ASCII? .. see the table at the bottom of this document. This is just for your reference. You should **not** need to use the actual ascii value in your code.

3) Continue with the previous exercise with the following additional rules:
   - Ask user for the first alphabet.
   - except every 5th letter, you must change it to lower case.
   - You can have more than one expression

   e.g. >Enter 1st letter: B

   Sample output:

   B,C,D,E,f,G,H,I,J,k,L,M,N,O,p,Q,R,S,T,u,V,W,X,Y,z

   e.g. >Enter 1st letter: C

   Sample output:

   C,D,E,F,g,H,I,J,K,l,M,N,O,P,q,R,S,T,U,v,W,X,Y,Z

4) Write a function which translates a word to a score, like Scrabble, except much fewer variation.
   - a,e,i,o,u,y will be worth 5 each.
   - q,x,z will be worth 20 each.
   - All other letters will be worth 10 each.
   - Any upper case will be worth 2 more.
   - Your output must list score for each letter as well.

   e.g. You entered: Enter the word: Ahha

   Sample output:

   Anna is worth 32 points (7 + 10 + 10 + 5)

Tips:

- E.g. char word[ ] = "Ahha";
- Your program should use "switch" statements to determine the score and display " Ahha gives 32 "
- Use intrinsic functions : tolower() and/or toupper(). E.g.
- #include <ctype.h>

```
…
word[i]=  tolower(word[i]);
// this will change a single character word[i] to lower case . If it is already lower case, nothing will change.
```

5) Given:

```
#define MAXR        25
#define MAXC       20

int abc[MAXR][MAXC];

void main()
{
…
int rows =  <write expression here to generate # value of 25 without using the MAXR> ;
int cols =  <write expression here to generate # value of 25 without using the MAXC>
```

Tips: sizeof(abc) == the capacity of the whole field.
sizeof(abc[0]) == the capacity of a single row
sizeof(abc[0][0]) == a single element

6) Use Enumeration Type to generate the following output (must read about Enumeration type at the beginning of this document first):

Sample Output Display:

```
North       == 0 degrees
NorthEast   == 45 degrees
East        == 90 degrees
…
West        == 270 degrees
NorthWest   == 315 degrees
```

Tip :.

```
for (i=N, i<=NW; i++)                          // where N and NW are part of an enum value
      printf(" %s == %d\n", directions[i] ,  go[i],);

       // e.g. directions[N] should contain "North", directions[NW] should contain "NorthWest".
```

7) Mr. Smith has 10 children. Each one was born 2 years apart. Just so happen that their gender alternate as well, i.e. boy, girl, boy, girl, etc. The oldest one is 30 years old, and is a daughter. Also, Mr. Smith is not so creative with names. He simply use "Beth" for girls, and "Dennis" for boys. Each additional child will simply have the same name attached with one of the vowels – a, e, i, o, u. This is how:

   - First daughter's name is Betha. The subsequent ones will be Bethe, followed by Bethi, Betho, and Bethu. - First son's name is Dennisa. The next one is Dennise… etc.
   - Display all of like the following:

```
Child Name Age
==============================
Daughter  Betha      30
Son       Dennisa    28
Daughter  Bethe      26
Son       Dennise    24
Daughter  Bethi      22
Son       Dennisi    20
Daughter  Betho      18
Son       Denniso    16
Daughter  Bethu      14
Son       Dennisu    12
```

> Note:
>
> Main body of your code cannot exceed 5 expressions (can be a compound expression). i.e. not counting main( ), headers, { , etc.
>
> You can even do this with one expression, ie. One line ended with ";".

Tip: design a few arrays to hold the a,e,i,o,u,, another variable array to hold Dennis and Beth, etc.,

8) Here is a block of cells with their corresponding slot number. Yellow area indicates the row and column numbers.

   The array can be: int cells[MAX_X][MAX_Y] where MAX_X = 4 and MAX_Y = 6

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24

   Write a function to print out the all values of within all cells like the table above. (You can skip the column numbers).

   Sample execution:

```
> Enter row# col#:  2, 5        ← this is from your input
Cell(2,5) = 18                  ← this is your output
```

9) Write a function to calculate factorial of a number: e.g. int factorial(int n).

10) Write a function to calculate fibonacci of a number: e.g. int fibonacci(int n).

11) Write the Sieve of Eratosthenes Prime number generator to generate all primes under N number which are entered as command line argument.

    To test:

    a) Display all primes under 100.
    b) Count # of primes under 1000000.

12) Write a function to produce the actual count of prime numbers under your input. (Utilize the Sieve of Eratosthenes)

    Sample test:

    Input : Max Count: 10000

    Output: There are 1229 prime numbers under 10000.

13) Write a very short program to do the following :

    10 bottles of drink on the wall, 10 bottles of drink!
    So take one down, pass it around, 9 more bottles of drink on the wall!
    9 bottles of drink on the wall, 9 bottles of drink!
    So take one down, pass it around, 8 more bottles of drink on the wall!
    8 bottles of drink on the wall, 8 bottles of drink!
    So take one down, pass it around, 6 more bottles of drink on the wall!
    …
    …
    1 bottle of drink on the wall, 1 bottle of drink!
    So take it down, pass it around, no more bottles of drink on the wall!

14) Write a program to do the following :

    20 bottles of drink on the wall, twenty-one bottles of drink!
    So take 1 down, pass it around, 19 more bottles of drink on the wall!
    90 bottles of drink on the wall, ninety-eight bottles of drink!
    So take 10 down, pass it around, 80 more bottles of drink on the wall!
    80 bottles of drink on the wall, eighty bottles of drink!
    So take 10 down, pass it around, 70 more bottles of drink on the wall!
    …
    …
    50 bottle of drink on the wall, fifty bottle of drink!
    So take it down, pass it around, 10 bottles of drink on the wall!

    Starts from 100, and stops at 40.

15) Move all your functions into a second file called "utility.cpp". Thus, you will have two separate files: one with the main() in it, another one has all these functions in it.

    To test, just call any function from your main() function.

16) Display a number in binary: void displayBinary(int number)
    i. Return : none
    ii. Parameters:

Int number : the number from user input

Sample test:

Input : Enter a base10 number : 16
16(base-10) = 10000(base 2)

17) Anagrams Set Problem. Write a function to display all the words and their matching anagram set number. For example:

Your words list:

char words[ ] [100] = { "aaabc", " cbaad", "aadbc", "cbaaa", "cdcdcd", "dccdcd", "abcaa" };

Your program displays:

aaabc	1
cbaad	2
aadbc	2
cbaaa	1
cdcdcd	3
abcaa	1
dccdcd	3