

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.

ARRAY

Take the construction like this:

<data type> variable[number of elements]
store homogeneous elements at contiguous locations

Simple example:

| | | | | |
|--|--|---|---|---|
| <p>Simple declaration and initialization for 1-D array</p> | <pre>int arr[5]; // data size of "int" * 5</pre> <table border="1" data-bbox="370 846 1489 1022"> <tr> <td data-bbox="370 846 607 1022"> <pre>for (i=0; i<5; i++) arr[i] = i*2;</pre> </td> <td data-bbox="613 846 1045 1022"> <p>Bad habit to hard code the "5". Use "sizeof(...)"</p> <pre>sizeof(int) == 4 sizeof(arr) = 4*5 # of elements = sizeof(arr) / sizeof(arr[0])</pre> </td> <td data-bbox="1052 846 1489 1022"> <pre>for (i=0; i< sizeof(arr) / sizeof(arr[0]); i++) arr[i] = <something>;</pre> </td> </tr> </table> <p>More samples: <pre>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.</pre></p> | <pre>for (i=0; i<5; i++) arr[i] = i*2;</pre> | <p>Bad habit to hard code the "5". Use "sizeof(...)"</p> <pre>sizeof(int) == 4 sizeof(arr) = 4*5 # of elements = sizeof(arr) / sizeof(arr[0])</pre> | <pre>for (i=0; i< sizeof(arr) / sizeof(arr[0]); i++) arr[i] = <something>;</pre> |
| <pre>for (i=0; i<5; i++) arr[i] = i*2;</pre> | <p>Bad habit to hard code the "5". Use "sizeof(...)"</p> <pre>sizeof(int) == 4 sizeof(arr) = 4*5 # of elements = sizeof(arr) / sizeof(arr[0])</pre> | <pre>for (i=0; i< sizeof(arr) / sizeof(arr[0]); i++) arr[i] = <something>;</pre> | | |
| <p>Pay attention to the boundary</p> | <p>e.g. int arr[4]; valid : arr[3] = 5; invalid : arr[4] = 5;</p> | | | |
| <p>Sample code</p> | <pre>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; }</pre> | | | |

| | |
|--|--|
| Simple declaration and initialization for 2D array | <pre>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}};</pre> |
|--|--|

ENUMERATION TYPE

| | |
|---|---|
| <p>Take the following macros:</p> <pre>#define NORTH 0 #define SOUTH 1 #define EAST 2 #define WEST 3</pre> <p>e.g. int wind_direction = NO;</p> | <p>Use Enumeration instead:</p> <pre>enum Directions { NORTH, SOUTH, EAST, WEST}; Directions dir; Valid : dir = NORTH; dir = WEST ; dir = (Directions)3; Bad : dir = (Directions) 4;</pre> |
| <p>use it with array</p> | <p>e.g. :</p> <pre>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</pre> |

TIME COMPLEXITY USING ARRAY:

| | |
|-----------------|---|
| Accessing Time: | O(1) [This is possible because elements are stored at contiguous locations] |
| Search Time: | O(n) for Sequential Search: O(log n) for Binary Search [If Array is sorted] |
| Insertion Time: | O(n) BUT Each insertion requires shifting all of the elements after the element! NOT Good! |
| Deletion Time: | O(n) BUT Each insertion requires shifting all of the elements after the element! NOT Good! |

FUNCTIONS

Take the construction like this:

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

Simple example:

| | |
|-----------|---|
| Example 1 | <pre>void doWork() { return; // does not return any value }</pre> |
| Example 2 | <pre>float sum(float data[], int num) { int i; float total=0.0; for (i=0 ; i<num; i++){ total = total + data[i]; } return total; }</pre> |
| Example 3 | <pre>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; }</pre> |
| Example 4 | <p>Use the Object struct sample above:</p> <pre>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; }</pre> |
| Example 5 | <pre>typedef struct Child { char name[7]; char gender; int age;</pre> |

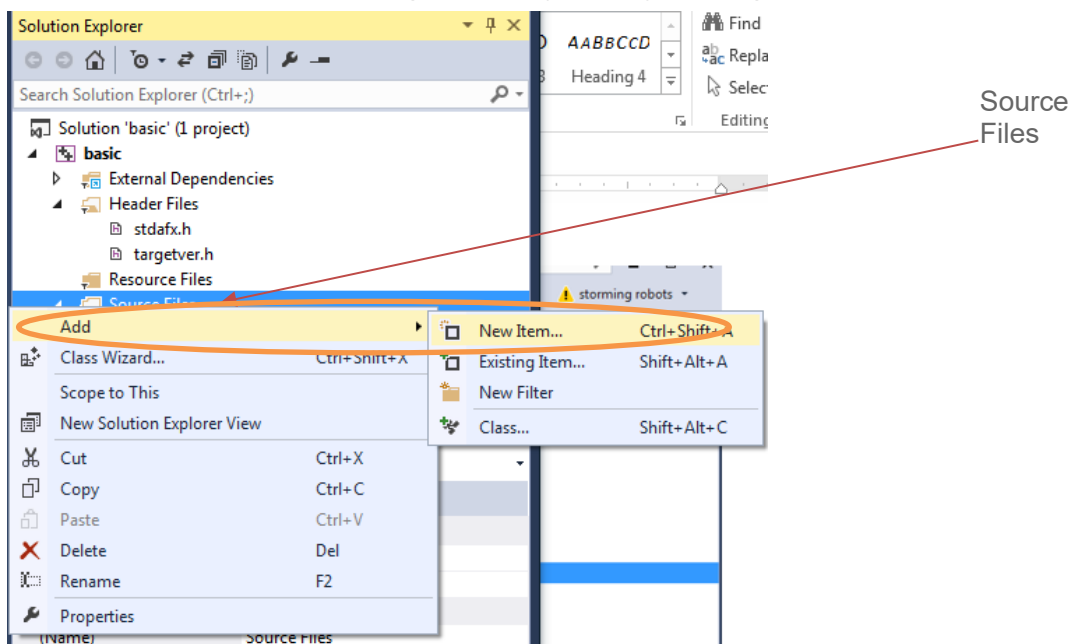
```

void setChild(int a, char s[], char g)
{
    strcpy( name, s, sizeof(name) );
    gender = g;
    age = a;
}
} Child;

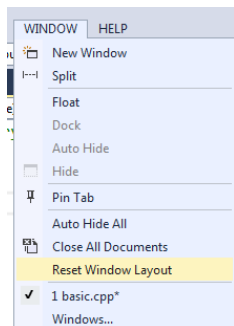
void main()
{
    Child ct;
    ct.setChild(10, 'F', 20);
    printf ("%d\n", ct.gender);
}
    
```

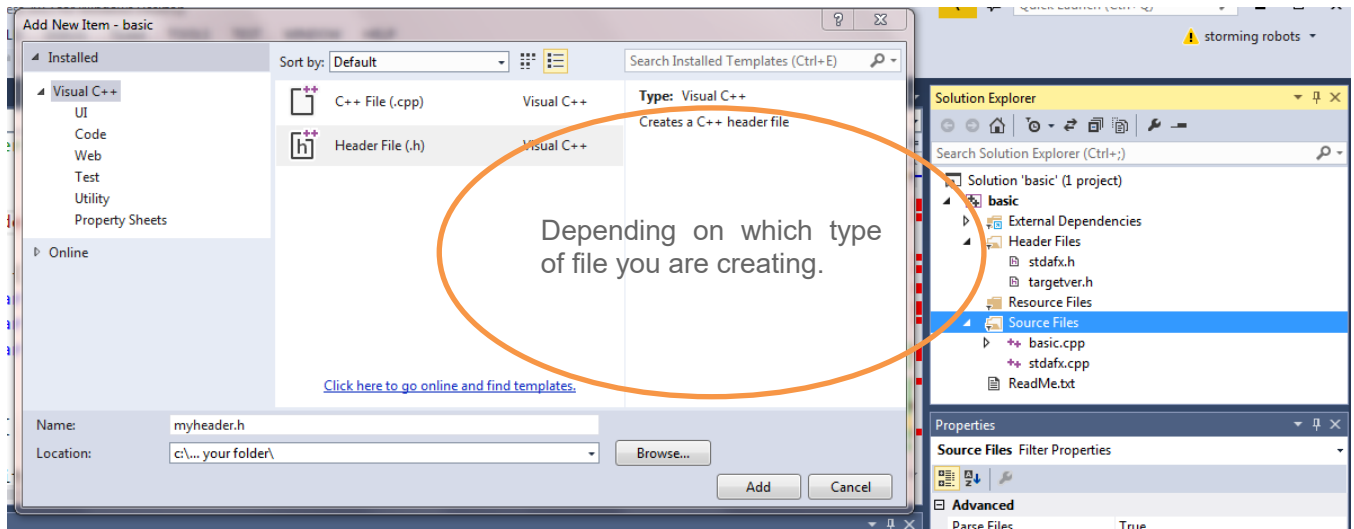
COLLECTION OF PROGRAMS AND HEADERS

How to add header and new programs into your project using Visual Studio Express:



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!

Tip: remember ASCII? .. see the table at the bottom of this document.

- 3) Write a program 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.
 - E.g. Ahha is worth 32 (7 + 10 + 10 + 5)

e.g. : char word[] = "Ahha";

Tips:

- 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.
```

- 4) Given:

```
#define MAXR    25  
#define MAXC    20
```

```
int abc[MAXR][MAXC];
```

```
void main()
```

```
{
```

```
...
```

```
int rows = <write expression here to generate the number of rows of abc> ;
```

```
int cols = <write expression here to generate the number of columns of abc>
```

```
// however, you cannot use "MAX", cannot hardcode any numeric constants...
```

```
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
```

5) Use the following code segment:

```
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".
```

to generate display:

```
North == 0 degrees
```

```
NorthEast == 45 degrees
```

```
East == 90 degrees
```

```
West == 270 degrees
```

```
NorthWest == 315 degrees
```

Tip : Review the Enumeration Type sample above.

- 6) 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.,

- 7) 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 program to print out the all values of within all cells like the table above. (You can skip the column numbers).

Sample output:

Cell (2,5) = 18

- 8) Write a program to calculate factorial of a number. Make this a function: `int factorial(int n)`.
- 9) Write a program to calculate fibonacci of a number. Make this a function: `int Fibonacci(int n)`.

10) 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.

11) Use Ex.9, ask user for an max count input. Your code will produce the actual count of prime numbers under your input.

Sample test:

Input : Enter the Max count: 10000

Output: total # of prime numbers = 1229

12) 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!

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

99 bottles of drink on the wall, ninety-nine bottles of drink!
So take one down, pass it around, ninety-eight more bottles of drink on the wall!
98 bottles of drink on the wall, ninety-eight bottles of drink!
So take one down, pass it around, ninety-seven more bottles of drink on the wall!
97 bottles of drink on the wall, ninety-seven bottles of drink!
So take one down, pass it around, ninety-six more bottles of drink on the wall!
...
...
...
20 bottle of drink on the wall, 20 bottle of drink!
So take it down, pass it around, twenty bottles of drink on the wall!

Starts from 99, and stops at 20.

Continue next page...

- 14) Create your own library file and header file containing functions and global variables, if any. Steps:
- i) Convert 3 programs into individual functions, including:
 - a) `int factorial(int input)`
 - b) `int fib(int count)`.
 - c) `int generatePrime(int)` :
 - i. Returns the actual value of the Nth prime number.
 - ii. Parameters:
 `int` : the Nth prime number, this is from user input
 - d) `int Scrabble(char [], int)`
 - i. Return : score value
 - ii. Parameters:
 `char []` : array of characters from user input
 `int` : the length of this array
 - ii) Move all your new data types, or global variables, and other header “`#include`” in a new header file, e.g. `myNewHeader.h` .
 - iii) Add these as new items into your “Source Files” list.
 - iv) Rebuild your project.
 - v) Prompt user which exercise he/she wants to run in your “main” routine. Test it. :

- 15) Display a number in binary: `void displayBinary(int number)`
- i. Return : none
 - ii. Parameters:
 `Int number` : the number from user input
 e.g. if `displayBinary(16)`, the output should look like 10000 .

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

Your words list:

```
char words[ ] [100] = { “aaabc”, “ cbaad”, “aadbc”, “cbaaa”, “cdcdd”, “dcccdd”, “abcaa” };
```

Your program displays:

| | |
|--------|---|
| aaabc | 1 |
| cbaad | 2 |
| aadbc | 2 |
| cbaaa | 1 |
| cdcdd | 3 |
| abcaa | 1 |
| dcccdd | 3 |

ASCII TABLE

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|-----|----|-----|------------------------------------|-----|----|-----|-------|-------|-----|----|-----|-------|-----|-----|----|-----|--------|-----|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 | | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | \$ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | (| (| 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 |) |) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [| [| 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 |] |] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | | DEL |