CPSC 259: Data Structures and Algorithms for Electrical Engineers

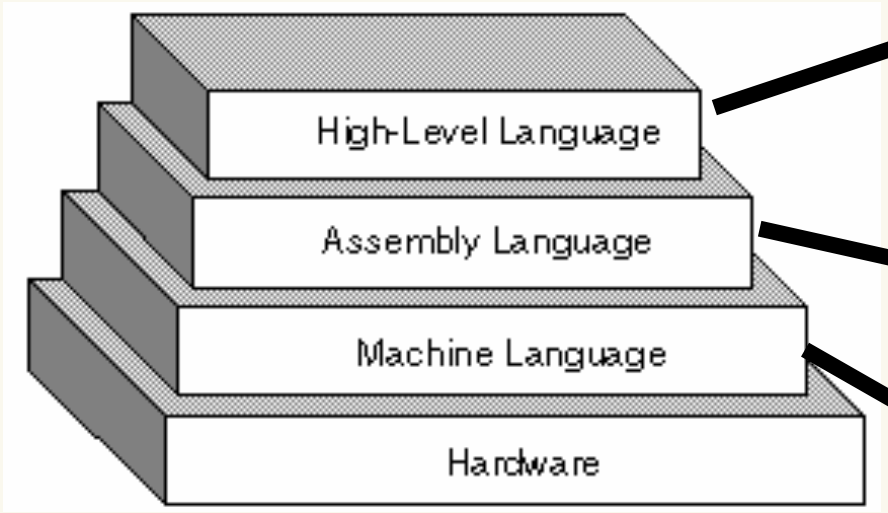
APSC 160 Review

Hassan Khosravi
Borrowing many questions from Ed Knorr

Learning Goal

- Briefly review some key programming concepts from APSC 160: functions, modularity, arrays

Review: Programming Languages




```
/* compute volume of cylinder */
volume = PI * radius * radius * height;

/* print volume on screen */
printf( "Volume of cylinder = %f\n", volume );

system("PAUSE");
return 0;
```

```
ldh r3, [r0], #2   ; load input
ldh r4, [r1], #2   ; load input
mul r5, r4, r3     ; multiply
add r2, r2, r5     ; accumulate
```

```
11101000 10110010 00000110 00000000 00000000 01011001 01101000 10100000
10110000 01000000 00000000 01101010 00000000 11101000 11000010 10010111
00000000 00000000 10100011 11011111 10110000 01000000 00000000 01101010
00000000 11101001 10001010 10010101 00000000 00000000 11101001 00101111
```

Question variable swap

Suppose that `var1` and `var2` are variables of type `int`. Which of the following code segments swaps the value of these two variables?

(a)
```
int temp = var1;
var1 = var2;
var2 = temp;
```

(b)
```
int temp = var1;
var2 = var1;
var1 = temp;
```

(c)
```
var1 = var2;
var2 = var1;
```

(d)
```
int temp1 = var1;
int temp2 = var2;
temp1 = temp2;
var2 = var1;
```

Question variable swap (answer)

Suppose that `var1` and `var2` are variables of type `int`. Which of the following code segments swaps the value of these two variables?

(a)
```
int temp = var1;
var1 = var2;
var2 = temp;
```

(b)
```
int temp = var1;
var2 = var1;
var1 = temp;
```

(c)
```
var1 = var2;
var2 = var1;
```

(d)
```
int temp1 = var1;
int temp2 = var2;
temp1 = temp2;
var2 = var1;
```

Question operator precedence

Assume that the following variable declarations have been made:

```
int a = 16;
int b = 4;
double c = 1.5;
```

What value is assigned to the variable ***d*** by the following statement?

```
double d = c + a * b;
```

(a)65.0

(b) 65.5

(c) 65

(d) 66

Question operator precedence (answer)

Assume that the following variable declarations have been made:

```
int a = 16;
int b = 4;
double c = 1.5;
```

What value is assigned to the variable ***d*** by the following statement?

```
double d = c + a * b;
```

(a)65.0

(b) 65.5

(c) 65

(d) 66

```
double d = 1.5 + (16 * 4);
```

Question variables and operators

Assume that the following variable declarations have been made:

```
int a = 16;
int b = 4;
double c = 1.5;
```

What value is assigned to the variable ***d*** by the following statement?

```
double d = b / a;
```

(a) 1

(b) 0

(c) 0.25

(d) 4

Question variables and operators (answer)

Assume that the following variable declarations have been made:

```
int a = 16;
int b = 4;
double c = 1.5;
```

What value is assigned to the variable **d** by the following statement?

```
double d = b / a;
```

(a) 1

(b) 0.0

(c) 0.25

(d) 4

Question Boolean logic

Suppose that variable "**t**" is a variable that has a value which evaluates to **true** and "**f**" is a variable that has a value which evaluates to **false**. Which one of the following expressions evaluates to **false**?

(a) `t && !f`

(b) `!t && f`

(c) `t || f`

(d) `!(t && f)`

(e) `t || ( !f && !t )`

Question Boolean logic (answer)

Suppose that variable "**t**" is a variable that has a value which evaluates to **true** and "**f**" is a variable that has a value which evaluates to **false**. Which one of the following expressions evaluates to **false**?

```
(a)   t && !f
(b)   !t && f
(c)   t || f
(d)   !(t && f)
(e)   t || ( !f && !t )
```

Reminder:
&& means "and"
|| means "or"
! means "not"

! is evaluated before || and &&

Question poor indentation

Consider the following poorly indented code segment: What are the values of **a**, **b** and **r** after this code segment has executed?

```
int r;
int a = -5;
int b =  6;
if( a < 0 || b > 0 )
  r = 1;
else
  r = 2;
  a = 0;
```

(a) a = 0 , b= 6 , r = 2
(b) a = 0 , b= 6 , r = 1
(c) a = -5 , b= 6 , r = undefined
(d) a = -5 , b= 6 , r = 2
(e) None of the above

Question poor indentation

Consider the following poorly indented code segment: What are the values of **a**, **b** and **r** after this code segment has executed?

```
int r;
int a = -5;
int b =  6;
if( a < 0 || b > 0 )
  r = 1;
else
  r = 2;
a = 0;
```

(a) a = 0 , b= 6 , r = 2

(b) a = 0 , b= 6 , r = 1

(c) a = -5 , b= 6 , r = undefined

(d) a = -5 , b= 6 , r = 2

(e) None of the above

Question looping

Consider the following code segment: What values do i and j have **after** this code segment has executed?

```
int i = 1;
int j = 0;

while( i < 5 && j < 4 ){
   j = j + i;
   i++;
}
printf( "i = %d, j = %d", i,j);
```

(a) i = 4 , j = 5
(b) i = 5 , j = 5
(c) i = 4 , j = 6
(d) i = 5 , j = 6
(e) None of the above

Question looping (answer)

Consider the following code segment: What values do i and j have **after** this code segment has executed?

i	j	statement
1	0	T
	1	
2		
		T
	3	
3		
		T
	6	
4		
		F

```
int i = 1;
int j = 0;

while( i < 5 && j < 4 ){
   j = j + i;
   i++;
}
printf( "i = %d, j = %d", i,j);
```

(c) i = 4, j = 6

Question looping

How many times is the printf statement executed in the following C code?

```
int x = 1;
while ( x < 15/4 ) {
   printf ("x = %d\n", x);
   x++;
}
```

A. never
B. once
C. twice
D. three times
E. four or more times (or infinite loop)

Question looping (answer)

How many times is the printf statement executed in the following C code?


```
int x = 1;
while ( x < 15/4 ) {
   printf ("x = %d\n", x);
   x++;
}
```

x	output
1	
	1
2	
	2
3	

A. never
B. once
C. twice
D. three times
E. four or more times (or infinite loop)

Question nested looping

Consider the following code segment: What values do `count1` and `count2` have after this code segment has executed?

(a) `count1=3,count2=8`
(b) `count1=3,count2=10`
(c) `count1=2,count2=8`
(d) `count1=2,count2=9`
(e) None of the above

```
int i;
int j;

int count1 = 0;
int count2 = 0;

for( i = 0; i < 3; i++ ) {
  count1++;
  for( j = 1; j < 4; j++ )
    count2++;
}
```

Question nested looping (answer)

Consider the following code segment: What values do `count1` and `count2` have after this code segment has executed?

first loop happens 3 times

second loop happens 3 times

(e) None of the above
Answer: count1= 3 count2=9

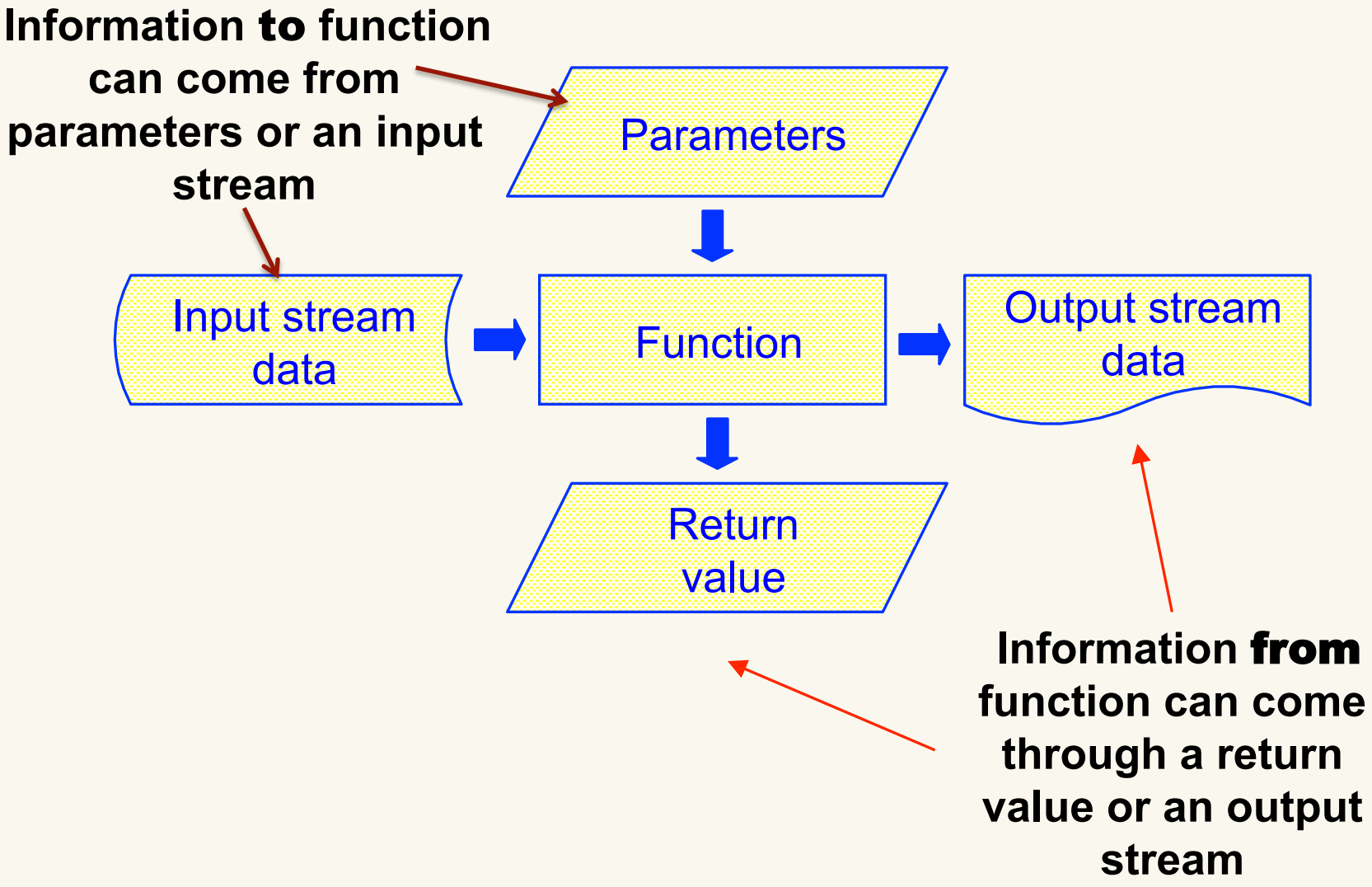
```
int i;
int j;

int count1 = 0;
int count2 = 0;

for( i = 0; i < 3; i++ ) {
  count1++;
  for( j = 1; j < 4; j++ )
    count2++;
}
```

Review: Modular Programming

- Imagine a 10,000-line program that consists only of one function: main.
  - Extremely difficult to **debug.**
  - Extremely difficult to **re-use** any part of the code.

- **Modularity**: A design principle used to manage the complexity of larger systems / programs.
  - Used in many engineering disciplines.
  - In software development, modularity can be implemented using **functions.**
    - We break our program into smaller modules (functions).

Functions review



Review: Function Parameters

- **Actual** parameter
  - Value(s) or variable(s) specified by the function *caller*
- **Formal** parameter
  - Variables found in the signature/header of the *function* itself
- Formal parameters must match with actual parameters in *order, number,* and *data type*.

What Happens When a Function is Called

1. Copy parameter values/addresses (if any) from the caller to the function, regardless of the variable names.
2. Execute the function. The function ends when we reach *any* return statement.
3. Pass back the answer (if any) via the return statement.
4. Destroy all local variables in the function (i.e., release/free memory).
   - (We'll start fresh if this function is ever called again.)
5. Return to the caller.
6. Finish the rest of the calling statement (after replacing the function call with the return value (if any)).

Review: Function (cont.)

```
/* Function to compute the maximum of two integers.
 PARAM:  a – one of the two integers
 PARAM:  b – the other of the two integers
 PRE:    NONE (no other assumptions)
 POST:   NONE (no side effects)
 RETURN: The larger of the two integers
 */
int compute_max(int a, int b){
  if (a > b)
    return a;
  return b;
}
```

Question function

Consider the following code segment: Fill in the blanks below to show what is output to the screen when this program runs?

```
void myFunc( int a, int b ){
  a = a + 4;
  b = b - 4;
  printf( "In myFunc a = %d b = %d\n", a, b );
}

int main(){
  int a = 5;
  int b = 7;
  myFunc( a, b );
  printf( "In main a = %d  b = %d\n", a, b );
  return 0;
}
```

In myFunc a =___ b =____

In main a =___ b =____

Review: Function Parameters (cont.)

- Parameters may be **passed/copied by value** ("call-by-value").
  - The value of the actual parameter is copied to the formal parameter when the function is called. For example, if `my_radius` is 2.0, then 2.0 is copied:

```
double compute_area( double radius );  /* prototype */
answer = compute_area( my_radius );  /* function call */
```

  - The actual parameters and formal parameters are different variables in memory, even if they are named the same.
  - If you change the value of the formal parameter, this does **not** affect the value of the actual parameter back in the caller's memory!

Question function (answer)

Consider the following code segment: Fill in the blanks below to show what is output to the screen when this program runs?

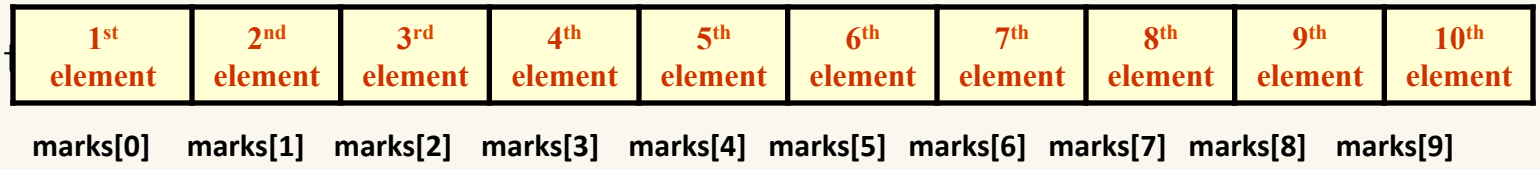
```
void myFunc( int a, int b ){
  a = a + 4;
  b = b - 4;
  printf( "In myFunc a = %d b = %d\n", a, b );
}

int main( void ){
  int a = 5;
  int b = 7;
  myFunc( a, b );
  printf( "In main a = %d  b = %d\n", a, b );
  return 0;
}
```

In myFunc a = 9 b = 3

In main a = 5 b = 7

Review: Arrays

- An array is a collection of similar data elements. These data elements have the same data type.
- The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the subscript). Declaring an array means specifying three things:
  - The data type: what kind of values it can store (int, char, float)
  - Name: to identify the array
  - The size: the maximum number of values that the array can hold
- Arrays are declared using the following syntax.



Elements of an array

- Accessing elements of an array
  - You can use a loop

```
int i, marks[10];
for(i=0;i<10;i++)
   marks[i] = -1;
```

  - Storing values in an array

Initialize the elements

int marks[]={90, 82, 78, 95, 88};

Inputting Values

```
int i, marks[10];
for(i=0;i<10;i++)
   scanf("%d",&marks[i]);
```

Assigning Values

```
for(i=0;i<10;i++)
  arr2[i] = arr1[i];
```

Question 1 - D array

Consider the following code segment. What is the value of sum after this code segment has executed?

```
int data[] = { 2, 4, 8, 16, 32, 64 };
int sum = 0;
int index = 1;

while( index < 4 ){
  sum += data[ index ];
  index++;
}
```

(a) sum = 30
(b) sum = 60
(c) sum = 32
(d) sum = 14
(e) None of the above

Question 1-D array (answer)

Consider the following code segment: What is the value of sum after this code segment has executed?

```
int data[] = { 2, 4, 8, 16, 32, 64 };
int sum = 0;
int index = 1;

while( index < 4 ){
  sum += data[ index ];
  index++;
}
```

Answer: 28

(a) sum = 30
(b) sum = 60
(c) sum = 32
(d) sum = 14
(e) None of the above

Question 1 - D array

Consider the following code function:

```
int doSomething( int data[], int numEls, int someVal ){
  int index,  found = -1;
  for( index = 0; index < numEls; index++ )
    if( data[ index ] == someVal )
      found = index;
  return found;
}
```

(a) It returns true if `someVal` is found in the first `numEls` entries of the array `data` and false otherwise

(b) If `someVal` is contained in the first `numEls` entries of the array `data`, it returns the value `someVal`, otherwise it returns -1.

(c) If `someVal` is contained in the first `numEls` entries of the array data, it returns the index of the last slot at which `someVal` is found, otherwise it returns -1.

(d) If `someVal` is contained in the first `numEls` entries of the array `data`, it returns the index of the first slot at which `someVal` is found, otherwise it returns -1.

Question 1 - D array

```
int doSomething( int data[], int numEls, int someVal ){
  int index,  found = -1;
  for( index = 0; index < numEls; index++ )
    if( data[ index ] == someVal )
      found = index;
  return found;
}
```

data	numEls	someVal	found	index	data[index]
12,12,89	3	12	-1	0	
			0		12
			1		12
					89

(c) If `someVal` is contained in the first `numEls` entries of the array `data`, it returns the index of the last slot at which `someVal` is found, otherwise it returns `-1`.

Question functions with 1-D array

Consider the following code segment: Fill in the blanks below to show what is output to the screen when this program runs?

```
#define SIZE 3
void process( int data[]);
int main( void ){
  int data[SIZE] = { 5, -1, 2 };
  int index;
  process( data );
  for( index = 0; index < SIZE; index++ )
    printf( "%d ", data[ index ] );
  return 0;
}

void process( int data[]){int index;
  for( index = 0; index < SIZE; index++ )
    data[ index ] = 0;
}
```

Answer: __ __ __

Review: Function Parameters (cont.)

- Sometimes, parameters are **passed/copied by reference** ("call-by-reference").

```
double getMaximum( double data[], int size );/* prototype */
answer = getMaximum( myArray, length ); /* function call */
```

  - The address (rather than the value) of the actual parameter is copied to the formal parameter when the function is called.
  - If you change the value of an element in an array using the formal parameter in the function, this changes the value (contents) of that memory location in the caller.

We'll talk more about this when we get to pointers

Question functions with 1-D array (answer)

Consider the following code segment: Fill in the blanks below to show what is output to the screen when this program runs?

```
#define SIZE 3
void process( int data[]);
int main( void ){
  int data[SIZE] = { 5, -1, 2 };
  int index;
  process( data );
  for( index = 0; index < SIZE; index++ )
    printf( "%d ", data[ index ] );
  return 0;
}

void process( int data[]){int index;
  for( index = 0; index < SIZE; index++ )
    data[ index ] = 0;
}
```

Answer: 0 0 0

Question finding min

Below are lines of code, ordered by length, that when rearranged can produce a snippet of code that, upon execution, will result in the variable "min" containing the minimum value in the array "x". Assume that all variables are declared as "int" and that array "x" contains SIZE elements of type int.

```
}
i++;
i = 1;
min = x[0];
min = x[i];
if (x[i] < min)
while (i < SIZE)
{
```

Question finding min (answer)

Below are lines of code, ordered by length, that when rearranged can produce a snippet of code that, upon execution, will result in the variable "min" containing the minimum value in the array "x". Assume that all variables are declared as "int" and that array "x" contains SIZE elements of type int.

```
}
i++;
i = 1;
min = x[0];
min = x[i];
if (x[i] < min)
while (i < SIZE)
{
```

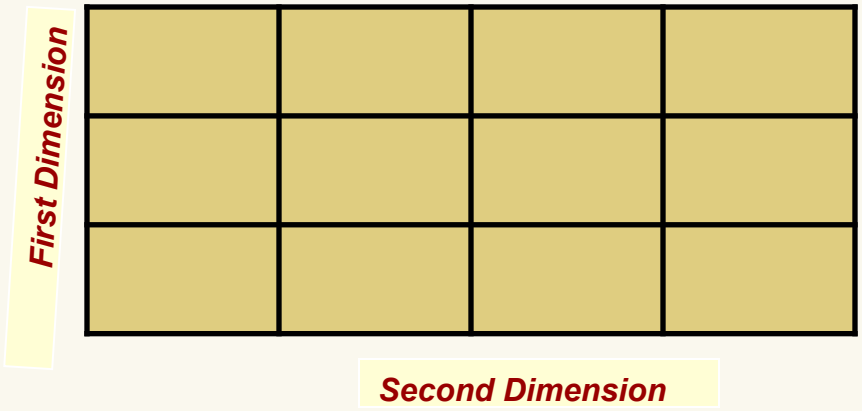
```
min = x[0];
i = 1;
while (i < SIZE)
{
  if (x[i] < min)
    min = x[i];
    i++;
}
```

Two dimensional arrays

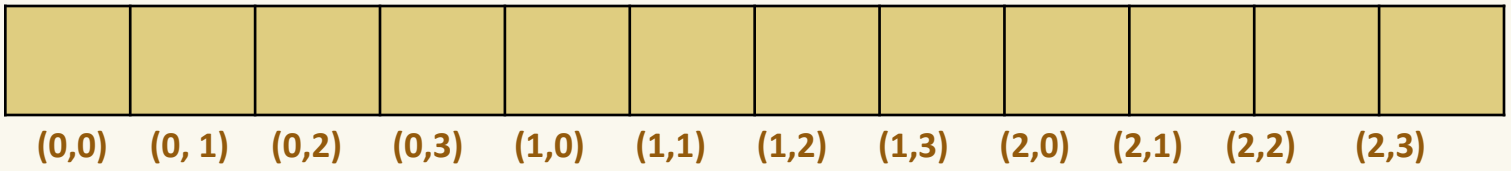
- A two dimensional array is specified using two subscripts where one subscript denotes row and the other denotes column.

```
dataType arrayName[rSize][cSize];
```

```
int marks[3][4];
```



- C stores a two dimensional array similar to a one dimensional array.



Question algorithm

Consider the following pseudocode :

```
read a number
while less then 100 integers read & the last number read is
not negative
do
    output the last number read
    read another number
end of the while loop
```

Suppose this program was modified to output the average of all the positive integers and the number of positive integers that exceeded the average. The simplest change to the program would:

(a) Not require an array or an additional loop.

(b) Require an array but no additional loop.

(c) Not require an array but am additional loop would be required.

(d) Require an array and at least one additional loop.

(e) None of the above

Question algorithm (answer)

```
read a number
while less then 100 integers read & the last number read is
not negative
do
    output the last number read
    read another number
end of the while loop
```

(d) Require an array and at least one additional loop.

{ one loop to read all values and find avg

{ one loop to compare each value with avg

Question algorithm

Consider the following pseudocode :

```
read a number
while less then 100 integers read & the last number read is
not negative
do
    output the last number read
    read another number
end of the while loop
```

Suppose the program given in the above pseudo code was instead modified to output the number of positive integers entered by the user, and their average. The simplest change to the program would:

(a) Not require an array or an additional loop.

(b) Require an array but no additional loop.

(c) Not require an array but am additional loop would be required.

(d) Require an array and at least one additional loop.

(e) None of the above

Question algorithm (answer)

```
read a number
while less then 100 integers read & the last number read is
not negative
do
    output the last number read
    read another number
end of the while loop
```

(a) Not require an array or an additional loop.

(b) Require an array but no additional loop.

(c) Not require an array but am additional loop would be required.

(d) Require an array and at least one additional loop.

(e) None of the above

Question loops and arrays

Suppose that an array called `arr1` is of type `int` and that the symbolic constant `SIZE` has been defined to represent the size of this array. Which one of the following pieces of code results in `arr2` becoming a copy of `arr1`?

(a)
```
int arr2[SIZE];
  arr2 = arr1;
```

(b)
```
int arr2[ SIZE ]  = arr1;
```

(c)
```
int arr2[ SIZE ]  = { arr1 };
```

(d) All of the above

(e) None of the above

Question loops and arrays (answer)

Suppose that an array called `arr1` is of type `int` and that the symbolic constant `SIZE` has been defined to represent the size of this array. Which one of the following pieces of code results in `arr2` becoming a copy of `arr1`?

(a)
```
int arr2[SIZE];
  arr2 = arr1;
```

(b)
```
int arr2[ SIZE ]  = arr1;
```

(c)
```
int arr2[ SIZE ]  = { arr1 };
```

(d) All of the above

(e) None of the above

Question Searching

N different positive integers are stored in the first N positions of an array. The first unused position in the array is indicated by the value -1

Suppose we want to look through the array to find the position of a positive integer "x". To achieve this functionality we would:

(a) Not require an additional array, or a loop.

(b) Require an additional array but no loop.

(c) Not require an additional array, but at least one additional loop would be required.

(d) Require an additional array and at least one additional loop.

(e) None of the above

Question Searching

N different positive integers are stored in the first N positions of an array. The first unused position in the array is indicated by the value -1

Suppose we want to look through the array to find the position of a positive integer "x". To achieve this functionality we would:

(a) Not require an additional array, or a loop.

(b) Require an additional array but no loop.

(c) Not require an additional array, but at least one additional loop would be required.

(d) Require an additional array and at least one additional loop.

(e) None of the above

Question Searching complexity

Now consider the situation where we have one array with N integers in it and one with 2N integers. We know that the number "x" is randomly located at some position in both arrays. (Note the position is different in each array.) On average the ratio of the amount of time (i.e the number of instructions executed) it takes to locate "x" in the array of size 2N when compared to the array of size N is:

(a) The same amount of time is needed)
(b) Twice as much time is needed to find "x" in the array of size 2N)
(c) Three times as much time is needed to find "x" in the array of size 2N
(d) Four times as much time is needed to find "x" in the array of size 2N
(e) Eight times as much time is needed to find "x" in the array of size 2N

Question Searching complexity (answer)

Now consider the situation where we have one array with N integers in it and one with 2N integers. We know that the number "x" is randomly located at some position in both arrays. (Note the position is different in each array.) On average the ratio of the amount of time (i.e the number of instructions executed) it takes to locate "x" in the array of size 2N when compared to the array of size N is:

$$N \rightarrow \quad \frac{1}{N}*1+\frac{1}{N}*2+\ldots\frac{1}{N}*N=\frac{(1+2+\ldots N)}{N}=\frac{\frac{N*(N+1)}{2}}{N}=\frac{N+1}{2}$$

$$2N \rightarrow \quad \frac{1}{2N}*1+\frac{1}{2N}*2+\ldots\frac{1}{2N}*2N=\frac{(1+2+\ldots 2N)}{2N}=\frac{\frac{2N*(2N+1)}{2}}{2N}=\frac{2N+1}{2}\approx N+1$$

(b) Twice as much time is needed to find "x" in the array or size 2N