C Programming – Lecture VI

Instructor Özgür ZEYDAN
http://cevre.beun.edu.tr/

Arrays

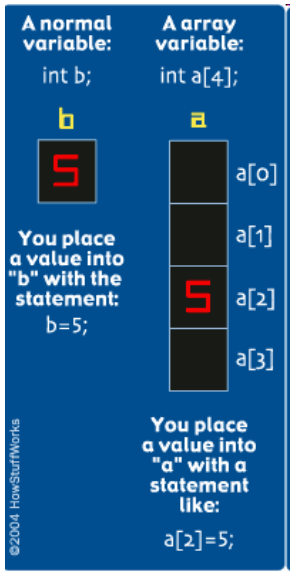
- Array is a collection of same type elements under the same variable identifier referenced by index number.
- Arrays are widely used within programming for different purposes such as sorting, searching and etc.
- Arrays allow you to store a group of data of a single type.
- Arrays can be created from any of the C data-types int, float, double or char.
- `<variable type> array_name[size_of_array];`

Arrays

- An array lets you declare and work with a collection of values of the same type. For example, you might want to create a collection of four integers. One way to do it would be to declare four integers directly:
- `int a, b, c, d;`
- This is okay, but what if you needed a thousand integers? An easier way is to declare an array of four integers:
- `int a[4];`
- The four separate integers inside this array are accessed by an **index**.
- All arrays start at index zero and go to n-1 in C.

Arrays

- Thus, `int a[4];` contains four elements.
- `int a[4];`
  - `a[0] = -18;`
  - `a[1] = 90;`
  - `a[2] = 5;`
  - `a[3] = 43;`



Arrays

- Using a loop to manipulate the index:

```
int a[4];
int i;
for (i=0; i<4; i++)
a[i] = 0;
```

Initializers

```
int x[ 5 ] = { 1, 2, 3, 4, 5 };
```

- If not enough initializers, rightmost elements become 0

  `int x[ 5 ] = { 0 }` (All elements 0)
- If too many initializers, a syntax error occurs
- C arrays have no bounds checking
- If size omitted, initializers determine it

  `int x[ ] = { 1, 2, 3, 4, 5 };`

  (5 initializers, therefore 5 element array)

Two Dimensional Arrays

- Tables with rows and columns (`m` by `n` array)
- Like matrices: specify row, then column
- Initialization

```
int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```

  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero

```
int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };
```

- Referencing elements
  - Specify row, then column

```
printf( "%d", b[ 0 ][ 1 ] );
```

Example – 1

- Write a C code to create one dimensional array with size 10.
- Computer will ask user to write integer values to that array.
- Finally, values of an array will be displayed on the screen.

Example – 1

```
#include <stdio.h>
#include <conio.h>
int main(void)
{ int i, a[10];
for (i=0; i<10; i++)
  { printf("Write %d. integer: ",i+1);
   scanf("%d",&a[i]);  }
for (i=0; i<10; i++)
   printf("%d. value: %d\n",i+1,a[i]);
getch();
return(0);   }
```

Example – 2

- Modify your first program to find the maximum and minimum of the integers given by user.

Example – 2

```
#include <stdio.h>
#include <conio.h>
int main(void)
{ int i, a[10], max, min;
max=0;
min=0;
for (i=0; i<10; i++)
  { printf("Write %d. integer: ",i+1);
    scanf("%d",&a[i]);
    if (a[i]>a[max]) max=i;
    if (a[i]<a[min]) min=i;   }
```

Example – 2

```
for (i=0; i<10; i++)
   printf("%d. value: %d\n",i+1,a[i]);
printf("\nMax value: %d\n",a[max]);
printf("\nMin value: %d\n",a[min]);
getch();
return(0);   }
```

Bubble Sort

- The simplest sorting algorithm is bubble sort.
- The bubble sort works by iterating down an array to be sorted from the first element to the last, comparing each pair of elements and switching their positions if necessary.
- This process is repeated as many times as necessary, until the array is sorted.

Bubble Sort - Animation

6 5 3 1 8 7 2 4

http://en.wikipedia.org/wiki/File:Bubble-sort-example-300px.gif

Bubble Sort Algorithm

```
for(int x=0; x<n; x++)
{   for(int y=0; y<n-1; y++)
    { if(a[y]>a[y+1])
        { temp = a[y+1];
          a[y+1] = a[y];
          a[y] = temp;
        }
     }
 }
```

Example – 3

- Write a C code that asks user to give values of two 3x3 matrices (Matrix A and matrix B).
- Then computer calculates matrix C which is equals to sum of two matrices.
- Finally computer displays C matrix.

- The sum A+B of two m-by-n matrices A and B is calculated entrywise:
- $C_{i,j} = A_{i,j} + B_{i,j}$, where $1 \le i \le m$ and $1 \le j \le n$.
- Example:

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 1+5 \\ 1+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 6 \\ 8 & 5 & 0 \end{bmatrix}$$

Example – 3

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a[3][3],b[3][3],c[3][3],i,j;
printf("Write integer values of A matrix\n");
for(i=0;i<3;i++)
 {for(j=0;j<3;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&a[i][j]);        }       }
printf("Write integer values of B matrix\n");
```

Example – 3

```
for(i=0;i<3;i++)
 {for(j=0;j<3;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&b[i][j]);        }       }
printf("C matrix\n");
for(i=0;i<3;i++)
 {for(j=0;j<3;j++)
  {c[i][j]=a[i][j]+b[i][j];
   printf("%4d",c[i][j]);        }
  printf("\n");  }
getch();        }
```

Example – 4

- Modify your previous C code that first asks dimensions (m and n) of A_{mxn} and B_{mxn} matrices.
  - Force user to write m and n values smaller than max size of matrices (for instance 20) and bigger than 0 by using **`do-while loop`**.
  - Be careful that m and n must be the same for all matrices to make addition.
- Next programs asks values of two matrices.
- Then computer calculates matrix C which is equals to sum of two matrices.
- Finally computer displays C matrix.

Example – 4

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a[20][20],b[20][20],c[20][20],i,j,m,n;
do{
printf("Write dimensions of matrices (m and n)(20 MAX): ");
scanf("%d %d",&m,&n);
} while(m<=0 || m>20 || n<=0 || n>20);
printf("Write integer values of A matrix\n");
for(i=0;i<m;i++)     {for(j=0;j<n;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&a[i][j]);       }       }
```

Example – 4

```
printf("Write integer values of B matrix\n");
for(i=0;i<m;i++)
 {for(j=0;j<n;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&b[i][j]);       }       }
printf("\nC matrix\n");
for(i=0;i<m;i++)
 {for(j=0;j<n;j++)
  {c[i][j]=a[i][j]+b[i][j];
   printf("%4d",c[i][j]);       }
  printf("\n"); }
getch();       }
```

Matrix Multiplication

- Multiplication of two matrices is defined only if the number of columns of the left matrix is the same as the number of rows of the right matrix.
- If A is an *m-by-n* matrix and B is an *n-by-p* matrix, then their matrix product AB is the *m-by-p* matrix whose entries are given by dot product of the corresponding row of A and the corresponding column of B:

$$[\mathbf{AB}]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \cdots + A_{i,n}B_{n,j} = \sum_{r=1}^{n} A_{i,r}B_{r,j}$$

- where $1 \le i \le m$ and $1 \le j \le p$
- *Source: http://en.wikipedia.org/wiki/Matrix_(mathematics)*

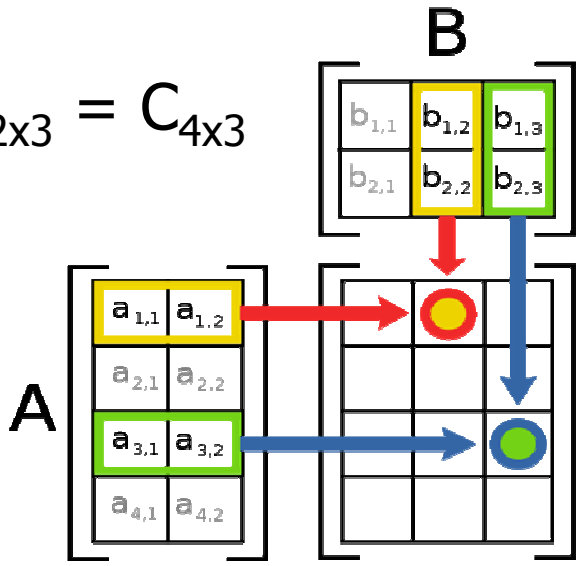
Matrix Multiplication

- For example, the underlined entry 2340 in the product is calculated as:
- (2 × 1000) + (3 × 100) + (4 × 10) = 2340

$$\begin{bmatrix} \underline{2} & \underline{3} & \underline{4} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \underline{1000} \\ 1 & \underline{100} \\ 0 & \underline{10} \end{bmatrix} = \begin{bmatrix} 3 & \underline{2340} \\ 0 & 1000 \end{bmatrix}.$$

Matrix Multiplication

$A_{4x2} \times B_{2x3} = C_{4x3}$



Matrix Multiplication Algorithm in C

```
for(i=0;i<row1;i++)
  {  for(j=0;j<column2;j++)
     { c[i][j]=0;
        for(k=0;k<column1;k++)
        {
    c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
     }
  }
```

Example – 5

- Write a C code that asks user to give dimensions (m, n and p) of A_{mxn} and B_{nxp} matrices.
  - Force user to write m and n values smaller than max size of matrices (for instance 20) and bigger than 0 by using **`do-while loop`**.
- Then computer calculates matrix C which is equals to multiplication of two matrices.
- Finally computer displays C matrix.

Example – 5

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a[20][20],b[20][20],c[20][20],i,j,k,m,n,p;
do{
printf("How many rows (m) in matrix A (20 MAX): ");
scanf("%d",&m);
printf("How many columns (n) in matrix A (20 MAX): ");
scanf("%d",&n);
printf("How many columns (p) in matrix B (20 MAX): ");
scanf("%d",&p);
} while(m<=0 || m>20 || n<=0 || n>20 || p<=0 || p>20);
```

Example – 5

```
printf("Write integer values of A matrix\n");
for(i=0;i<m;i++)
 {for(j=0;j<n;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&a[i][j]);       }       }
printf("Write integer values of B matrix\n");
for(i=0;i<n;i++)
 {for(j=0;j<p;j++)
  {printf("Value of %d. row and %d. column: ",i+1,j+1);
   scanf("%d",&b[i][j]);       }       }
```

Example – 5

```
printf("\nC matrix\n");
for(i=0;i<m;i++)
{ for(j=0;j<p;j++)
  { c[i][j] = 0;
    for(k=0;k<n;k++)
    { c[i][j] = c[i][j] + a[i][k] * b[k][j];
    }
    printf("%6d",c[i][j]);
  }
  printf("\n");
}
getch();        }
```

Homeworks

- Write a C code that sorts the values of one dimensional array by using "Bubble Sort Algorithm".
- Write a C code that calculates and displays determinant of a square matrix.
- Write a C code that calculates and displays transpose of a matrix.