
EEE146
ARRAYS

Motivation

- You may need to define many variables of the same type.
  - Defining so many variables one by one is cumbersome.
- Probably you would like to execute similar statements on these variables.
  - You wouldn't want to write the same statements over and over for each variable.

Example #1

- Write a program that reads the grades of 300 students in EEE145 and finds the average.

Example #1 *(cont'd)*

- Soln:

```
#include <iostream>
using namespace std;

int main()
{  int i, sum=0, grade; float avg;

   for (i=0; i<300; i++)
   {  cin>>grade;
      sum += grade;
   }
   avg = sum/300.0;
   cout<<"Average = " <<avg<<endl;
   return 0;
}
```

This was simple.
Since we don't need to store all values, taking the sum is enough.
So, we don't need an array.

Example #2

- Write a program that reads the grades of 300 students in EEE145 and finds those that are below the average.

Example #2 *(cont'd)*

- Soln #1:

```
#include <iosteam>
using namespace std;

int main()
{  int i, sum=0, grade; float avg;

   for (i=0; i<300; i++)
   {  cin>>grade;
      sum += grade;
   }
   avg = sum/300.0;
   for (i=0; i<300; i++)
      if (grade<avg)
        cout<<"Below avg: "
   return 0;
}
```

WRONG!
"grade" contains the score of the last student.
You have already lost the previous 299 scores.

Example #2 *(cont'd)*

- Soln #2:

```
#include <iostream>
using namespace std;

int main()
{   int i,sum,gr0,gr1,gr2,...,gr349;
    float avg;

    cin>>gr0;
    cin>>gr1;
    cin>>gr2;
    ...

    sum = gr0+gr1+gr2+...+gr349;
    avg = sum/300.0;
    if (gr0<ave)
       cout<<"Below avg: " << gr0 << endl;
    if (gr1<ave)
       cout<<"Below avg: " << gr1 << endl;
    if (gr2<ave)
       cout<<"Below avg: " << gr2 << endl;

    ...
    if (gr299<ave)
       cout<<"Below avg: " << gr299 << endl;
return 0;
}
```

You cannot skip these with "..." You have to repeat each of these statements 300 times.

What is still wrong here?

Example #2 *(cont'd)*

- Soln #3:

```
#include <iostream>
using namespace std;
int main()
{  int i, sum=0, grade[300]; float avg;

   for (i=0; i<300; i++)
   {  cin >> grade[i];
      sum += grade[i];
   }
   avg = sum/300.0;
   for (i=0; i<300; i++)
      if (grade[i]<avg)
       cout<<"Below avg: "<<grade[i]<<endl;
   return 0;
}
```

Defines an array consisting of 350 integer values.
In the definition, the value in the brackets is the number of elements (size).

This means the *i*th element of the array. Here, the value in the brackets is the index, not the size.

Arrays

- An array is a variable that is a collection of multiple values of the same type.
- Syntax:

```
type array_name[int_constant_value]={initializer_list};
```

- The size has to be of int type and must be a fixed value (i.e., known at compile time).
- You can define an array of any type (eg: int, float, enum student_type, etc.)
- All elements of the array have the same type.
- You cannot use the {} format for initialization after variable definition,

ie, `int a[3]={5,8,2}` is correct, but

```
int a[3];
...
a={5,8,2}
```

is wrong.

Arrays

- The index must of int type.

```
int k[5];
k[k[4]/k[1]]=2; /* Correct as long as k[4]/k[1] is nonnegative*/
k[1.5] = 3;     /* Error since 1.5 is not int */
```

Arrays

- The lower bound must be nonnegative.

```
float m[8];    int i;

m[-2] = 9.2;  /* Syntax error */

i=-2;
m[i] = 9.2;  /* Run-time error */
```

Initializing Arrays

- The elements of a local array are arbitrary (as all other local variables).
- The elements of a global array are initialized to zero by default (as all other global variables).

Initializing Arrays

- You may initialize an array during definition as follows:

```
int array[5] = {10, 8, 36, 9, 13};
```

- However, you cannot perform such an initialization after the definition, i.e.,

```
int array[5];
array = {10, 8, 36, 9, 13};
```

is syntactically wrong.

Initializing Arrays

- If the number of initializers is less than the size of the array:
  - initialization starts by assigning the first value to the first element and continues as such,
  - remaining elements are initialized to zero (even if the array was local)
- Eg: For the definition

```
int array[5] = {10, 8, 36};
```

the first 3 elements get the values 10, 8, and 36, respectively. array[3] and array[4] become 0.

Initializing Arrays

- If the number of initializers is more than the size of the array, it is a syntax error.
- It is also possible to skip the size of the array iff the array is explicitly initialized.
  - In this case, the compiler fills in the size to the number of initializers.
  - Eg: For the definition

```
int array[ ] = {5, 9, 16, 3, 5, 2, 4};
```

the compiler acts as if the array was defined as follows:

```
int array[7] = {5, 9, 16, 3, 5, 2, 4};
```

Example #3

- Read 100 integers and find the unbiased variance.

```
#include<iostream>
using namespace std;
int main()
{   int X[100], i;
    float avg=0,var=0;

    for (i=0; i<100; i++)
    {   cin >> X[i];
        avg += X[i];
    }
    avg /= 100;
    for (i=0; i<100; i++)
       var += (X[i]-avg) * (X[i]-avg);
    var /= 99;
    cout<<"variance: " << var << endl;;
    return 0;
}
```

Unbiased variance of a sample is defined as

$$\frac{\sum_{i=1}^{N}(X_i-\mu)^2}{N-1}$$

Example #4

- Find the histogram of the scores in Midterm 1.

```
#include <iostream>
using namespace std;
int main()
{  int i, hist[101]={0}, score;

   for (i=0; i<300; i++)
   {  cin >> score;
      hist[score]++;
   }
   for (i=0; i<101; i++)
      cout<<hist[i]<<" student(s) got "<< i <<endl;
   return 0;
}
```

Example #5

- Check if the array in the input is symmetric (eg: 8, 10, 6, 2, 6, 10, 8)

```
#include <iostream>
#define SIZE 10
using namespace std;

int main()
{  int numbers[SIZE], i;

   for (i=0; i<SIZE; i++)
      cin>>numbers[i];
   for (i=0; i<SIZE/2; i++)
      if (numbers[i] != numbers[SIZE-1-i])
         break;
   cout<< "It is ";
   if (i!=SIZE/2)
      cout<<"not ";
   cout<<"symmetric"<<endl;
   return 0;
}
```

Arrays Have Fixed Size!

- The size of an array must be stated at compile time.
- This means you cannot define the size when you run the program. You should fix it while writing the program.
- This is a very serious limitation for arrays. Arrays are not fit for dynamic programming.
  - You should use pointers for this purpose.

Arrays Have Fixed Size!

- What you can do is to define very large arrays, making the size practically infinite
  → Wastes too much memory.
- Your program may exceed the maximum memory limit for the process.

Arrays as Parameters

- Although you write like a value parameter, an array is always passed by reference (variable parameter).
  - Therefore, when you make a change in an element of an array in the function, the change is visible from the caller.

Example #7

- Fill in an array of integer from input.

```
#include <iostream>
using namespace std;
void read_array(int ar[10])
{  int i;
   for (i=0; i<10; i++)
      cin >> ar[i];
}
int main()
{  int a[10], i;
   read_array(a);
   for (i=0; i<10; i++)
      cout<< a[i]<< " ";
   return 0;
}
```

Arrays as Parameters

- The size you specify in the function header is not important; you may even skip it.
- Eg:

```
void func(int arr[5])
{  int i;
   for (i=0; i<10; i++)
      arr[i]=i;
}
int main()
{  int a[10], i;
   func(a);
   for (i=0; i<10; i++)
      cout<< a[i]<< " ";
   return 0;
}
```

- This will work without any problems though the function header is misleading.

Example #8

- Write a function that inverts its array parameter.

```
void invert(int ar[10])
{   int i, temp;
    for (i=0; i<10; i++)
    {   temp=ar[i];
        ar[i] = ar[9-i];
        ar[9-i] = temp;
    }
}
```

What is wrong here?

Example #9: Bubble Sort

- Sort the values in an array in ascending order.

```
#include <iostream>
using namespace std;
void read_array(int ar[], int size)
{  int i;
   for (i=0; i<size; i++)
      cin>> ar[i];
}

void print_array(int ar[], int size)
{  int i;
   for (i=0; i<size; i++)
      cout<<  ar[i] << " ";
   cout<<"\n";
}

void swap(int *a, int *b)
{  int temp;
   temp = *a;
   *a = *b;
   *b = temp;
}
```

```
void bubble_sort(int ar[], int size)
{  int i, j;
   for (i = 0; i < size; i++)
      for (j = i + 1; j < size; j++)
         if (ar[i] > ar[j])
            swap(&ar[i],&ar[j]);
}

int main()
{  int ar[10];
   read_array(ar,10);
   bubble_sort(ar,10);
   print_array(ar,10);
   return 0;
}
```

Multi-dimensional arrays

- If the problem implies a physical context of several dimensions, a multidimensional array may be used.
  - Chairs in a classroom
  - Rooms on different floors of a building
  - Coordinates on a map
  - Coordinates in space
  - A timetable (hours versus days)

2D Arrays

- In a classroom where the chairs are organized as a grid of 8 rows and 10 columns

```
int chair[8][10];
for (i=0; i<8; i++)
    for (j=0; j<10; j++)
        cin>>chair[i][j];
```

Example #1

- Construct a student's timetable. Read course code (assume all courses have distinct codes).

```
int table[8][5];
for (i=0; i<8; i++)
    for (j=0; j<5; j++)
        cin>> table[i][j];
```

Example #2

- Store a map of every pixel on the screen (256 colors/pixel). Assume a resolution of 1024x768.

```
unsigned char screen[1024][768];
```

Example #3

- Read the number of inhabitants in every flat in a building. Assume there are 10 floors with 5 flats in every floor. Find the flats that have occupancy above the average.

Example #3

```
int flat[10][5], i, j, sum=0;
float avg;

for (i=0; i<10; i++)
   for (j=0; j<5; j++)
   {  cin>> flat[i][j];
      sum += flat[i][j];
   }
avg = sum/50.0;
for (i=0; i<10; i++)
   for (j=0; j<5; j++)
      if (flat[i][j]>avg)
         cout<<"On floor  "<<i<<" in flat "<<j;
```

Example #4

- Mark every soldier with "1" on a map. Rest is all zeros.
- Find the coordinates of all soldiers that can reach a given coordinate in 10 steps.

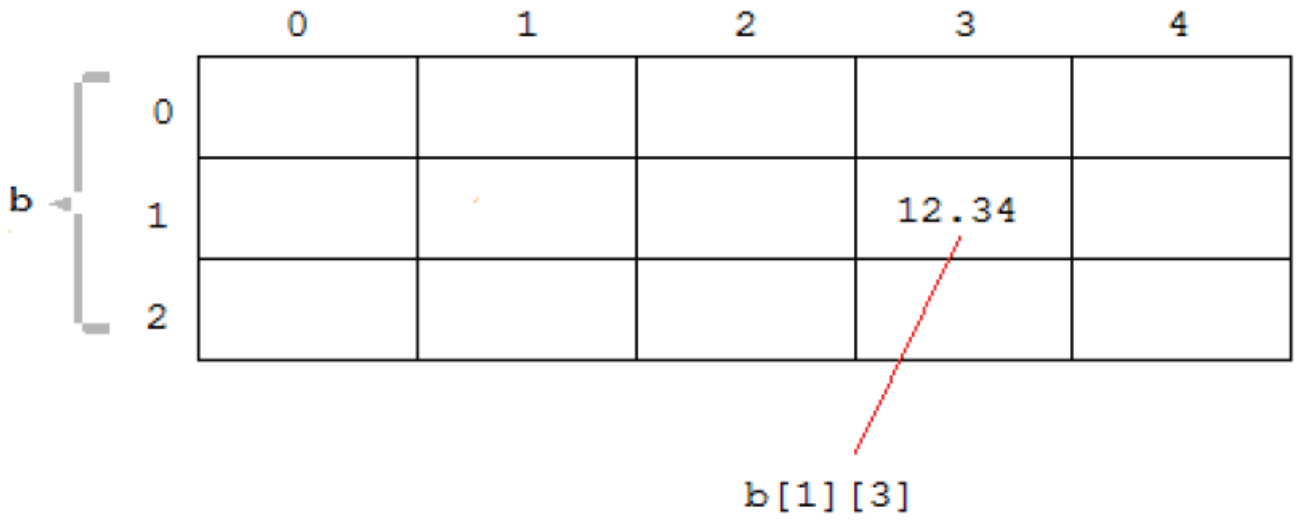
Example #4

```
#define ABS(x) ((x)<0) ? -(x) : (x)
int map[1000][1000], int coord_x, coord_y, i, j;

for (i=0; i<1000; i++)
   for (j=0; j<1000; j++)
      cin>> map[i][j];
cin>> coord_x >> coord_y; /* Read coordinates*/

for (i=coord_x-10; i<=coord_x+10; i++)
   for (j=coord_y-10; j<=coord_y+10; j++)
      if (map[i][j])
         if ((ABS(coord_x-i)+ABS(coord_y-j) <= 10)
            cout<< i <<"  "<< j;
```

Multidimensional Arrays

```
double a[5];      // 5-element one-dimensional array
float b[3][5];    // 15-element two-dimensional array
int c[5][4][10]; // 200-element three-dimensional ar.
```

```
b[1][3] = 12.34;
```



```
int x[3][4] = {  11, 34, 42, 60,
                 72, 99, 10, 50,
                  0, 66, 21, 38};
```

or

```
int x[3][4] = {  {11, 34, 42, 60},
                 {72, 99, 10, 50},
                 { 0, 66, 21, 38} };
```

equivalent to the matrix:

$$x = \begin{bmatrix} 11 & 34 & 42 & 60 \\ 72 & 99 & 10 & 50 \\ 0 & 66 & 21 & 38 \end{bmatrix}$$

Passing Arrays to Functions

```
#include <iostream>
using namespace std;
// returns the sum of first n elements
double sum(double x[], int n) {
  double t = 0.0;
  for(int i=0; i<n; i++){
     t = t + x[i];
  }
  return t;
}

int main () {
  double a[5], s;
  cout << "Enter 5 reals: ";
  for (int k=0; k<5; k++) cin >> a[k];

  s = sum(a, 5);
  cout << "sum of the elements is " << s << endl;
}
```

```
Enter 5 reals: 1.1 2.2 3.3 4.4 5.5
sum of the elements is 16.5
```

Homework

1. Write a function to calculate the trace of a square matrix.
2. Write a function to find the dot product of two vectors.
3. Write a function that calculate the transpose of a matrix.
4. Write a program that inputs a matrix, then finds the maximum element together with its row and column index.