ARRAYS

.

What is an array?

- A linear arrangement of data of the same type of elements
- An array has to be declared first with the maximum number of elements it can store
- int marks[100];
- char name[20];

How is an array stored?

- Starting from a given memory location, the successive array elements are allocated space in consecutive memory locations.

Array a

  - x: starting address of the array in memory
  - k: number of bytes allocated per array element
- a[i] □ is allocated memory location at address $x + i*k$

Index Rule

- An array index must evaluate to an int between 0 and n-1 where n is the number of elements in the array.
  - marks[76]
  - marks[i*2+k] // provided i*2+k is between 0 1nd 99

C Array bounds are not checked

~~#define S 100~~
~~marks[S] = 10;~~

```
if (0<=i && i<100)
    marks[i] = 10;
else printf ("Array index %d
            out of range", i);
```

Use

- An array element can be used wherever a simple variable of the same type can be used.
  - Examples :

```
scanf ("%d", &marks[i]);
marks[i] = (int) (sqrt(21.089));
```

Things you can and can't do

- You can not
  - use = to assign one array variable to another
  - use == to directly compare array variables
  - directly scanf or printf arrays
- But you can do these things on array elements.
- You can write functions to do them.

Averaging marks

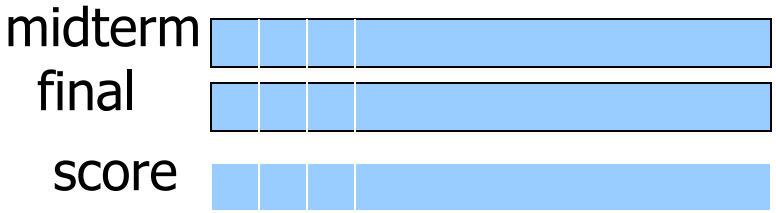
```
#define CLASS_SIZE 50
double marks[CLASS_SIZE] ;
double total=0.0;
int i;
printf (“Enter %d grades \n”, CLASS_SIZE);
for (i=0; i<CLASS_SIZE; i++)
      scanf(“%f”, &marks[i]);
for (i=0; i<CLASS_SIZE; i++)    {
      printf (“ %d . %f\n”,i, marks[i]);
      total += marks[i] ;
}
printf (“Average = %f\n”, total / (double) CLASS_SIZE) ;
```

- Are Arrays necessary to solve the above problem ?
- What about this problem :
  - read student marks, print all marks above average only.

```
#define MtWeight 0.3
#define FinalWeight 0.7
#define MaxStudents 100

int NumStudents;
int midterm[MaxStudents];
int final[MaxStudents];
double score[MaxStudents];
```

Parallel Arrays




```
/* Suppose we have input the value of NumStudents,
   read student i's grades for midterm and final, and
   stored them in midterm[i] and final[i]
   Store a weighted average in the array score */
if (NumStudents < MaxStudents)
    for (i=0; i<NumStudents; i++)    {            score[i]
   = MtWeight* (double) midterm[i] +
                        FinalWeight* (double) final[i];
}
```

Reading Array Elements

```
/* Read in student midterm and final grades and store
   them in two arrays */
#define MaxStudents 100
int midterm[MaxStudents], final[MaxStudents];
int NumStudents ;     /* actual no of students */
int i, done, Smidterm, Sfinal;

printf ("Input no of students :");
scanf("%d", &NumStudents) ;
if (NumStudents > MaxStudents)
    printf ("Too many students") ;
else
    for (i=0; i<NumStudents; i++)
     scanf("%d%d", &midterm[i], &final[i]);
```

Reading Arrays - II

```
/* Read in student midterm and final grades and store them in 2 arrays */
#define MaxStudents 100
int midterm[MaxStudents], final[MaxStudents];
int NumStudents ;     /* actual no of students */
int i, done, Smidterm, Sfinal;
done=FALSE; NumStudents=0;
while (!done)    {
    scanf("%d%d", &Smidterm, &Sfinal);
    if (Smidterm !=-1 || NumStudents>=MaxStudents)
     done = TRUE;
    else       {
     midterm[NumStudents] = Smidterm;
     final[NumStudents] = Sfinal;
     NumStudents++;
    }
}
```

Size of an array

- How do you keep track of the number of elements in the array ?
  - 1. Use an integer to store the current size of the array.

```
#define MAX 100
int  size;
float cost[MAX] ;
```

  - 2. Use a special value to mark the last element in an array. If 10 values are stored, keep the values in cost[0], ... , cost[9], have cost[10] = -1
  - 3. Use the 0th array element to store the size (cost[0]), and store the values in cost[1], ... , cost[cost[0]]

Add an element to an array

1. cost[size] = newval; size++;
2. for (i=0; cost[i] != -1; i++) ;
   cost[i] = newval;
   cost[i+1] = -1;
3. cost[0]++;
   cost[cost[0]] = newval;

Arrays as parameters of functions

- An array passed as a parameter is not copied

Array operations

```
#define MAXS 100
int insert (int[], int, int, int) ;
int delete (int[], int, int) ;
int getelement (int[], int, int) ;
int readarray (int[], int) ;
main () {
    int a[MAXS];
    int size;
    size = readarray (a, 10) ;
    size = insert (a, size, 4, 7) ;
    x = getelement (a, size, 3) ;
    size = delete (a, size, 3) ;
}
```

Array operations

```
#define MAXS 100
int insert (int[], int, int, int) ;
int delete (int[], int, int) ;
int getelement (int[], int, int) ;
int readarray (int[], int) ;
main () {
    int a[MAXS];
    int size;
    size = readarray (a, 10) ;
    size = insert (a, size, 4, 7) ;
    x = getelement (a, size, 3) ;
    size = delete (a, size, 3) ;
}
```

```
int readarray (int x[], int size)    {
    int i;
    for (i=0; i<size; i++)
    scanf("%d", &x[i]) ;
    return size;
}
```

```
int getelement (int x[], int size, int pos){
    if (pos <size) return x[pos] ;
    return -1;
}
```

```
int insert (int x[], int size, int pos. int val){
    for (k=size; k>pos; k--)
      x[k] = x[k-1] ;
    x[pos] = val ;
    return size+1;
}
```

}

```
void reverse (int x[],
              int size)  {



}
```

```
int findmax (int x[], int
    size)        {

}
```

```
    for (i=1; i< size; i++)
     if (x[i] > max)
          max = x[i] ;
    return max;
}
```

```
void reverse (int x[], int size)
    int i;
    for (i=0; i< (size/2); i++)
     temp = x[size-i-1] ;
     x[size-1-1] = x[i] ;
     x[i] = temp;
}
```

```
int findmax (int x[], int size) {
    int i, max;
    max = x[0];
    for (i=1; i< size; i++)
     if (x[i] > max)
          max = x[i] ;
    return max;
}
```

Strings

- Strings are 1-dimensional arrays of type char.
- By convention, a string in C is terminated by the end-of-string sentinel \0, or null character.
- String constant : “abc” is a character array of size 4, with the last element being the null chaaracter \0.
- char s[] = “abc” ;

a	b	c	\0