Arrays and Strings

DSA I

SY (COMP)

COEP Technological University

----SHRIDA KALAMKAR

Example: Find Minimum of Numbers

3 numbers

```
int a, b, c;
if ((a <= b) && (a <= c))
        min = a;
else if (b <= c)
        min = b;
else
        min = c;

printf("Minimum=%d",min);
```

4 numbers

```
int a, b, c, d;
if ((a <= b) && (a <= c) && (a <= d))
        min = a;
 else if ((b <= c) && (b <= d))
        min = b;
else if (c <= d)
        min = c;
else
        min = d;
printf("Minimum no is=%d",min);
```

The Problem

- Suppose we have 10 numbers to handle.
- Or 20.
- Or 100.
- Where do we store the numbers ?
- Use 100 variables ??
- How to tackle this problem?

- Solution:
  - Use arrays.
- Construct one variable (called array or subscripted variable) capable of storing or holding all the hundred values.

Introduction to Arrays

- **Array is a data structure which can store a fixed size sequential collection of elements/data items of same data type.**
- An array is a collective name given to a group of 'similar quantities'.
- What is important is that the quantities must be 'similar'
  - percentage marks of 100 students
  - or salaries of 300 employees,
  - or ages of 50 employees.
- These similar elements could be all ints, or all floats, or all chars, etc.
- Usually, the **array of characters is called a 'string'**, whereas an array of int or float is called simply an array.

Introduction to Arrays

- An array is a collection of similar data type value in a single variable.
- It is a **derived data type** in C, which is constructed from fundamental data type of C language.
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- For example if you want to store 10 numbers then instead of defining 10 variables its easy to define an array of 10 length.
- In the C programming language an array can be One-Dimensional, Two-Dimensional and Multidimensional.

Declaring Arrays

- Like variables, the arrays that are used in a program must be declared before they are used.
- **General syntax:**

type array_name [size];

– **type** specifies the type of element that will be contained in the array. **type** can be any valid C data type (int, float, char, etc.)

– size is an integer constant which indicates the maximum number of elements that can be stored inside the array. The **size** must be an integer constant greater than zero.

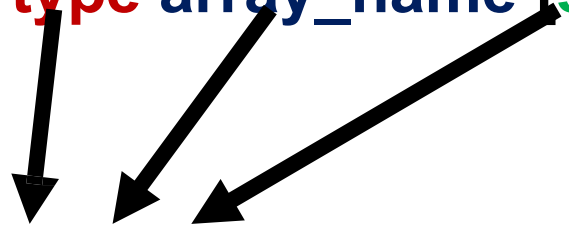
int marks[5];

– **marks is an array which can store maximum 5 elements of integer type.**

Declaring Arrays contd…

- **Examples**:

```
type array_name [size];
```



```
int x[10];
char line[80];
float points[150];
char name[35];
```

Declaring Arrays contd…

- **If we are not sure of the exact size of the array, we can define an array of a large size.**

  int marks[50];

  though in a particular run we may only be using, say, 10 elements.

- In declaration, if size of array is not mention then compiler will give an error.

  int arr[]; //error

Initializing Arrays

- Initializing is a process to initialize the value in array variable.
- **General form:**

  type array_name[size] = { list of values };

- **Examples:**

  int marks[5] = {72, 83, 65, 80, 76};

or

```
int marks[5];
marks[0] = 72;
marks[1] = 83;    // like this for all elements
……
```

```
char name[5] = {‘A’, ‘m’, ‘i’, ‘t’,’\0’};
```

or

```
char name[5] = “Amit”;
```

Initializing Arrays

- In initialization, the size of an array may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

  int flag[ ] = {1, 1, 1, 0};

  char name[ ] = {'A', 'm', 'i', 't', '\0'};

- If the number of values in the list is less than the number of elements, the remaining elements are automatically set to zero.

  float total[5] = {24.2, -12.5, 35.1};

  total[0]=24.2, total[1]=-12.5, total[2]=35.1, total[3]=0, total[4]=0

- All individual array elements that are not assigned explicit initial values will automatically be set to zero.

- The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

Initializing Arrays

```
char text [ ] = "Data Structures";
```

- This declaration will cause text to be an 16-element character array. The first 15 elements will represent the 15 characters within the word California, and **the 16th element will represent the null character (\0)which is automatically added at the end of the string.**

- The declaration could also have been written

```
char text [16] = "Data Structures";
```

- If array size is specified as 10 in the above statement then the program works without any warning/error in C

```
char text [15] = "Data Structures";
```

the characters at the end of the string (in this case, the null character) will be lost.

Initializing Arrays

- When character array is initialized with comma separated list of characters and array size is not specified, **compiler doesn't create extra space for string terminator '\0'**

```
char name[ ] = {'A', 'm', 'i', 't'};
```

- If the size is too large, e.g.

```
char text[20] = "California";
```

the extra array elements may be assigned zeros, or they may be filled with meaningless characters.

Accessing Array Elements

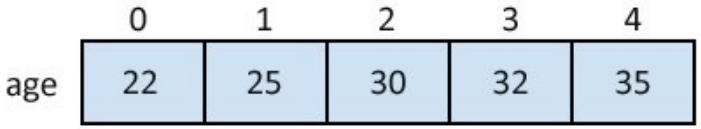
- **A particular element of the array can be accessed by specifying two things:**
  - **Name of the array.**
  - **Index (relative position) of the element in the array.**

- This is done by placing the index of the element within square brackets after the name of the array.
- For example

```
int age[5] = {22,25,30,32,35};
int var = age[3];
```

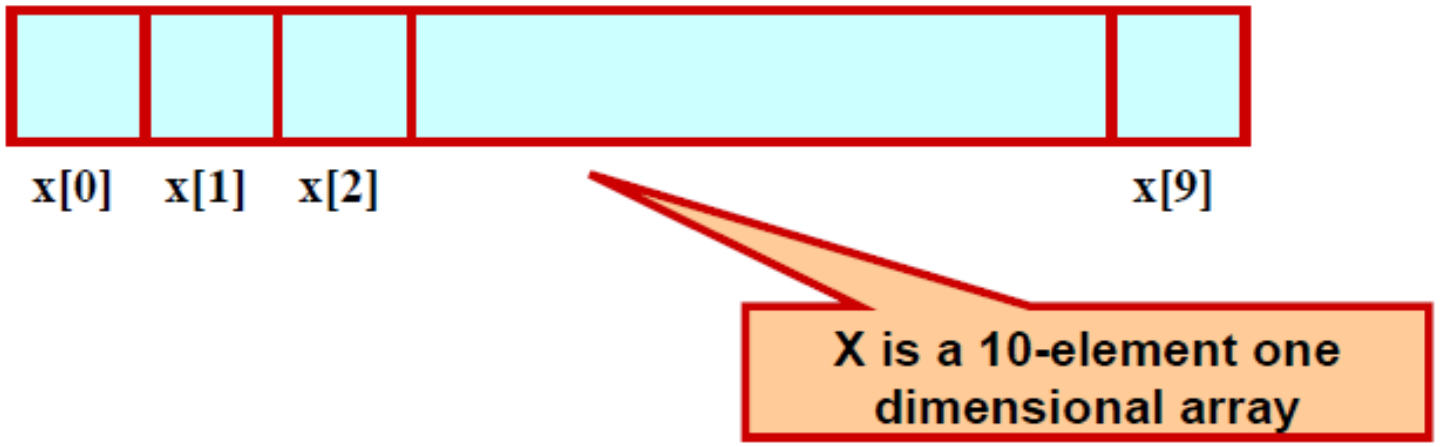
It will take the 4th element from the array 'age' and assign the value to 'var' variable.

Note- Indexing of elements begins with 0 and not with 1.



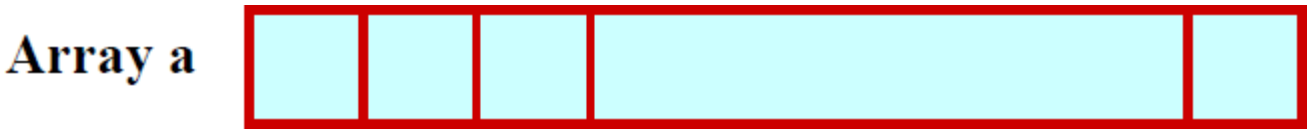
Accessing Array Elements

- All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1.
- **Example:**
  - **An array is defined as int x[10];**
  - **The first element of the array x can be accessed as x[0], fourth element as x[3], tenth element as x[9], etc.**



How an array is stored in memory?

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



- **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**

For example- **int a[10]; starting address is 1020**

then a[2] is allocated memory location at address 1020 + 2*4

i.e. 1028

A Warning

- If we specify the size of array as ‘N’ then we can access elements upto ‘N-1’ but in C if we try to access elements after ‘N-1’ i.e Nth element or N+1th element then we does not get any error message.

- Process of Checking the extreme limit of array is called **Bound Checking** and C does not perform **Bound Checking.**

- If the array range exceeds then we will get garbage value as result.

A Warning

- In C, while accessing array elements, array bounds are not checked.
- Example:

```
#include<stdio.h>
int main() {
        int a[5];
        printf("%d",a[7]);
}
```

- The above assignment would not necessarily cause an error.
- Rather, it may result in unpredictable program results.
- Here array size specified is 5.
- So we have access to following array elements – a[0],a[1],a[2],a[3] and a[4]
- But accessing a[5] or a[6] etc. causes Garbage Value to be used.
  - **IT IS PROGRAMMERS RESPONSIBILITY TO NOT EXCEED ARRAY BOUNDS/LIMITS**

Note

- Error: Constant Expression Require

```
#include<stdio.h>
void main() {
int i=10;
int a[i];
}
```

- We have declared an array whose size is equal to the **value of variable**.
- If we changed the value of variable then array size is going to change.
- i is initialized to 10 and using **a[i] does not mean a[10]** because ‘i’ is Integer variable whose value can be changed inside program.

How to enter(input) the elements into array?

- **You cannot directly scanf or printf arrays**

```
scanf("%d", &a);
printf ("%d", a);
```

- One element at a time

```
scanf ("%d", &a[1]);
scanf ("%d", &a[2]);
.....
scanf ("%d", &a[25]);
```

- Using loop

```
for (j=0; j<25; j++)
        scanf ("%d", &a[j]);
```

- The ampersand (&) is necessary.

How to print the elements of an array?

- By printing them one element at a time.

```
for (j=0; j<25; j++)
        printf ("\n %d", a[j]);
```

  – The elements are printed one per line.

```
printf ("\n");
for (j=0; j<25; j++)
        printf (" %d", a[j]);
```

  – The elements are printed all in one line (starting with a new line).

- Array element reading from the keyboard

```
scanf("%d",&a[0]);          scanf("%d",&a[1]);
```

- Array display on the output screen.

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

Accessing One Dimensional Array Elements

```
#<include<stdio.h>
void main()   {
  int i, a[10];
  printf (“Enter the array elements”);
  for (i=0;i<10;i++) {
     scanf(“%d”,&a[i]);
   }
   printf(“Entered array elements are”);
   for(i=0;i<10;i++)   {
       printf(“%d \n”,a[i]);
    }
 }
```

```
1 2 3 4 5 6 7 8 9 10
Entered array elements are
1
2
3
4
5
6
7
8
9
10
```

OUTPUT

Enter the array elements

Example-declaration, assignment and accessing arrays

```
#include <stdio.h>
int main () {
  int n[ 10 ]; // array declaration
  int i,j;
  for ( i = 0; i < 10; i++ ) {
    n[ i ] = i + 100;
  }
  for (j = 0; j < 10; j++ ) {
    printf("Element[%d] = %d\n", j, n[j] );
   }
  return 0;
}
```

Output:

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Example- Write a program to find average marks obtained by a class of 30 students in a test.

Things you can't do

- **You cannot use = to assign one array values to another**

  a = b; /* a and b are arrays */

  -But assign values of an array to another array, element by element

  a[i]=b[i+2]; a[i]=a[i+1];

a =1,2,3,4,5,6,7,8,9,10

b1,3,5,7,9,2,4,6,8,10

- **You cannot use == to directly compare array variables**

  if (a = = b) ……..

  - But can compare two arrays, element by element :

  a[i]==b[i+1];

- **You cannot directly scanf or printf arrays**

  printf ("……", a);

Things you can't do

For example-

```
int array[11];
printf("Write down your ID number!\n");
scanf("%d", array);  // not allowed
```

```
// correct way
int array[11];
printf("Write down your ID number!\n");
for(int i=0;i<id_length;i++)
      scanf("%d", &array[i]);
```