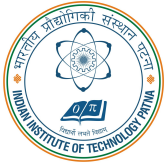
Array



Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Array

- Many applications require multiple data items that have common characteristics
  - In mathematics, we often express such groups of data items in indexed form:
    - $x_1, x_2, x_3, \ldots, x_n$
- Array is a data structure which can represent a collection of data items which have the same data type (`float`/`int`/`char`/...)

Example: Printing number in reverse

3 numbers

```
int a,b,c;
scanf("%d",&a);
scanf("%d",&b);
scanf("%d",&c);
printf("%d",c);
printf("%d",b);
printf("%d\n",a);
```

4 numbers

```
int a,b,c;
scanf("%d",&a);
scanf("%d",&b);
scanf("%d",&c);
scanf("%d",&d);
printf("%d",d);
printf("%d",c);
printf("%d",b);
printf("%d\n",a);
```

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**

Printing in reverse using arrays

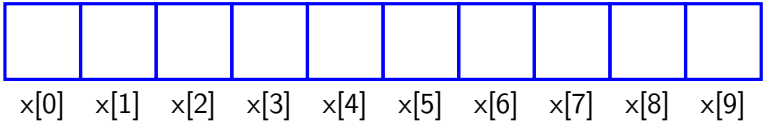
```
void main(){
  int n,A[100],i;
  printf("How many numbers to read?");
  scanf("%d",&n);
  for(i=0;i<n;i++)
    scanf("%d",&A[i]);
  for(i=n-1;i>=0;i--)
    printf("%d ",A[i]);
  printf("\n");
}
```

Using arrays

- All the data items constituting the group share the same name

  int x[10]

- Individual elements are accessed by specifying the index



Simple example

```
void main(){
  int data[10],i;
  for(i=0;i<10;i++)
    data[i]=i;
  i=0;
  while(i<10){
    printf("data[%d]=%d\n",data[i]);
    i++;
  }
}
```

Declaring arrays

- Like variables, the arrays 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 (int, float, char, etc.)
  - `size` is an integer constant which indicates the maximum number of elements that can be stored inside the array

```
int marks[5];
```

  - `marks` is an array that can store a maximum of 5 integers

Examples

```
int x[10];
char line[100];
float points[90];
char names[35];
```

- 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

Accessing array elements

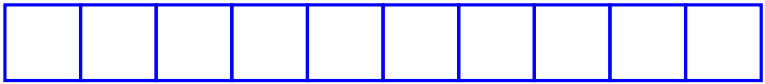
- A particular element of the array can be accessed by specifying two things:
  - Name of the array
  - Index (relative position) of the element in the array
- In C, the index of an array starts from zero
- Example:
  - An array is defined as int x[10];
  - The first element of the array x can be accessed as x[0], fourth element as x[3], tenth element as x[9], etc.
- The array index must evaluate to an integer between 0 and n-1 where n is the maximum number of elements possible in the array

```
a[x+2] = 25; b[3*x-y] = a[10-x] + 5;
```

- Remember that each array element is a variable in itself, and can be used anywhere a variable can be used (in expressions, assignments, conditions,...)

Storage for array 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
- x[i] — is allocated memory location at address x+i*k

Storage

```
void main(){
  int data[10],i;
  for(i=0;i<10;i++)
    printf("&data[%d]=%u\n",i,&data[i]);
}
```

```
&data[0]=227818948
&data[1]=227818952
&data[3]=227818956
...
```

Initialization of arrays

- General form:

```
type array_name[size] =  list of values ;
```

- Examples:

```
int marks[5] = {72, 83, 65, 80, 76};
char name[4] = {'A', 'm', 'i', 't'};
```

- The size 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';}
```

How to read the elements of an array?

- By reading them one element at a time

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

- The ampersand (&) is necessary
- The elements can be entered all in one line of in different lines

Warning

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

```
int marks[5];
⋮
⋮
marks[8] = 75;
```

- The above assignment would not necessarily cause an error
- Rather, it may result in unpredictable program results

Reading into an array

```
void main(){
  const int CONST_SIZE=100;
  int i,size;
  float marks[CONST_SIZE],total;
  scanf("%d",&size);
  for(i=0,total=0;i<size;i++){
      scanf("%f",&marks[i]);
      total=total+marks[i];
    }
  printf("Total=%f, Avg=%f\n",total,total/size);
}
```

How to print the elements of an array?

- By printing them one element at a time, one per line

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

- The elements are printed all in one line

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

How to copy the elements of an array to another?

- By copying individual elements

```
for(j=0;j<25;j++)
  a[j]=b[j];
```

- The elements assignments will follow the rules of assignments expressions
- Destination array must have sufficient size

Find the minimum of 10 numbers

```
void main(){
  int i,min,a[10];
  for(i=0;i<10;i++)
    scanf("%d",&a[i]);
  min=a[0];
  for(i=1;i<10;i++){
    if(a[i]<min)
      min=a[i];
  }
  printf("Min=%d\n",min);
}
```

Alternate version - 1

```
const int size=10;
void main(){
  int i,min,a[size];
  for(i=0;i<size;i++)
    scanf("%d",&a[i]);
  min=a[0];
  for(i=1;i<size;i++){
    if(a[i]<min)
      min=a[i];
  }
  printf("Min=%d\n",min);
}
```

Change only one line to change the problem size

Alternate version - 2

```
#define size 10
void main(){
  int i,min,a[size];
  for(i=0;i<size;i++)
    scanf("%d",&a[i]);
  min=a[0];
  for(i=1;i<size;i++){
    if(a[i]<min)
      min=a[i];
  }
  printf("Min=%d\n",min);
}
```

Change only one line to change the problem size

Used `define` macro

`#define` **macro**

- `#define X Y`
- Preprocessor directive
- Compiler will first replace all occurrences of string X with string Y in the program, then compile the program
- Similar effect as read-only variables (const), but no storage allocated
- We prefer you use const instead of `#define`

Alternate version - 3

```
void main(){
  int i,min,a[100],n;
  scanf("%d",&n); /* no of elements */
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  min=a[0];
  for(i=1;i<n;i++){
    if(a[i]<min)
      min=a[i];
  }
  printf("Min=%d\n",min);
}
```

Define an array of large size and use only the required number of elements

Computing CPI

```
const int nsub=6;
void main(){
  int grade[nsub],credit[nsub],i,gpsum=0,creditsum=0;
  double cpi;
  scanf("%d",&n); /* no of elements */
  for(i=0;i<n;i++)
    scanf("%d%d",&grade[i],&credit[i]);
  for(i=0;i<n;i++){
    gpsum += grade[i]*credit[i];
    creditsum += credit[i];
  }
  cpi=((double)gpsum)/creditsum;
  printf("CPI=%lf\n",cpi);
}
```

Handling two arrays

Binary search

- Searching for an element k in a sorted array A with n elements
- Idea:
  - Choose the middle element A[n/2]
  - If k == A[n/2], we are done
  - If k < A[n/2], search for k between A[0] and A[n/2 -1]
  - If k > A[n/2], search for k between A[n/2 + 1] and A[n-1]
  - Repeat until either k is found, or no more elements to search
- Requires less number of comparisons than linear search in the worst case ($\log_2 n$ instead of n)

Binary search

```
void main(){
  int A[100],n,k,i,mid,low,high;
  scanf("%d%d",&n,&k); /* no of elements */
  for(i=0;i<n;i++) scanf("%d",&A[i]);
  low=0; high=n-1; mid=low+(high-low)/2;
  while(high>=low){
    printf("low=%d,high=%d,mid=%d,A[%d]=%d\n",low,high,mid,mid,A[mid]);
    if(A[mid]==k){
      printf("%d is found\n",k); break;
    }
    if(k<A[mid]) high=mid-1;
    else low=mid+1;
  }
  if(high<low) printf("%d not found\n",k);
}
```

Binary search

```
8 21
9 11 14 17 19 20 23 27
low=0 high=7 mid=3 A[3]=17
low=4 high=7 mid=5 A[5]=20
low=6 high=7 mid=6 A[6]=23
21 not found
```

```
8 14
9 11 14 17 19 20 23 27
low=0 high=7 mid=3 A[3]=17
low=4 high=2 mid=1 A[1]=11
low=2 high=2 mid=2 A[2]=14
14 is found
```

Selection sort

- Sort the elements of an array A with *n* elements in ascending order
- Idea:
  - Find the min of the n elements, swap it with A[0] (so min is at A[0] now)
  - Now find the min of the remaining n-1 elements, swap it with A[1] (so 2nd min is at A[1] now)
  - Continue until no more elements left

Selection sort

```
void main(){
  int A[100],n,k,i,j,min,pos,temp;
  scanf("%d",&n); /* no of elements */
  for(i=0;i<n;i++) scanf("%d",&A[i]);
  for(i=0;i<n-1;i++){
    min=A[i]; pos=i;
    for(j=i+1;j<n;j++){
      if(A[j]<min){
        min=A[j]; pos=j;
      }
    }
    temp=A[i];   A[i]=A[pos];   A[pos]=temp;
  }
  for(k=0;k<n;++k) printf("%d ",A[k]);
  printf("\n");
}
```

Selection sort

6

7 12 5 15 17 9

5 12 7 15 17 9

5 7 12 15 17 9

5 7 9 15 17 12

5 7 9 12 17 15

5 7 9 12 15 17

8

9 8 7 6 5 4 3 2

2 8 7 6 5 4 3 9

2 3 7 6 5 4 8 9

2 3 4 6 5 7 8 9

2 3 4 5 6 7 8 9

2 3 4 5 6 7 8 9

2 3 4 5 6 7 8 9

2 3 4 5 6 7 8 9

Things you cannot do

- Use = to assign one array variable to another

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

- Use == to directly compare array variables

```
if (a = = b) ..........
```

- directly scanf or printf arrays

```
printf ("......", a);
```

Character arrays and strings

```
char C[6] = { 'r', 'o', 'h', 'i', 't', '\0' };
```

- C[0] gets the value 'a', C[1] the value 'b', and so on. The last (6th) location receives the null character '\0'
- Null-terminated (last character is '\0') character arrays are also called strings
- Strings can be initialized in an alternative way. The last declaration is equivalent to:

```
char C[8] = "rohit";
```

- The trailing null character is missing here. C automatically puts it at the end if you define it like this
- Note also that for individual characters, C uses single quotes, whereas for strings, it uses double quotes

Reading strings: %s format

```
void main(){
  char name[25];
  scanf("%s",name);
  printf("Name=%s\n",name);
}
```

- %s reads a string into a character array given the array name or start address. It ends the string with '\0'

An example

```
#define SIZE 25
void main(){
  int i,count=0;
  char name[25];
  scanf("%s",name);
  printf("Name=%s\n",name);
  for(i=0;name[i]!='\0';i++)
    if(name[i]=='a') count++;
  printf("%d\n",count);
}
```

Palindrome checking

```
#define SIZE 25
void main(){
  int i,flag,count=0;
  char name[25];
  scanf("%s",name);
  printf("Name=%s\n",name);
  for(i=0;name[i]!='\0';i++);
  printf("Total length=%d\n",i);
  count=i; flag=0;
  for(i=0;i<count;i++)
    if(name[i]!=name[count-i-1]) flag=1;
  if(flag==0) printf("%s is a palindrome\n",name);
  else  printf("%s is NOT a palindrome\n",name);
}
```

Some exercises

- Write a C program that reads an integer n and stores the first n Fibonacci numbers in an array.
- Write a C program that reads an integer n and uses an array to efficiently find out the first n prime numbers.
- Read in an integer n, read in n integers and print the integer with the highest frequency.
- Read in an integer n, read in n numbers and find out the mean, median and mode.
- Read in two names and compare them and print them in lexicographic (dictionary) order.
- Read in an integer n, read in n names and print the last name when compared in lexicographic order.