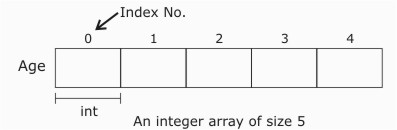
**Array**

An array is a collection of data elements of same data type. It is described by a single name and each element of an array is referenced by using array name and its subscript no.

**Declaration of Array**

Type arrayName[numberOfElements];

**For example,**  
int Age[5] ;  
float cost[30];



**Initialization of One Dimensional Array**

An array can be initialized along with declaration. For array initialization it is required to place the elements separated by commas enclosed within braces.  
int A[5] = {11,2,23,4,15};  
It is possible to leave the array size open. The compiler will count the array size.  
int B[] = {6,7,8,9,15,12};

**Referring to Array Elements**

In any point of a program in which an array is visible, we can access the value of any of its elements individually as if it was a normal variable, thus being able to both read and modify its value. The format is as simple as:  
name[index]  
  
**Examples:**   
cout<<age[4];      //print an array element  
age[4]=55;         // assign value to an array element  
cin>>age[4];       //input element 4

**Using Loop to input an Array from user**

int age [10], i ;  
for (i=0 ; i<10; i++)  
{  
  cin>>age[i];  
}

**Arrays as Parameters**

At some moment we may need to pass an array to a function as a parameter. In C++ it is not possible to pass a complete block of memory by value as a parameter to a function, but we are allowed to pass its address.   
For example, the following function:   
void print(int A[])  
accepts a parameter of type "array of int" called A.   
In order to pass to this function an array declared as:  
int arr[20];  
we need to write a call like this:   
print(arr);

**Here is a complete example:**

#include <iostream>  
using namespace std;   
void print(int A[], int length)  
{  
  for (int n=0; n<length; n++)  
    cout << A[n] << " ";  
  cout << "\n";  
}

int main ()  
{  
  int arr[] = {5, 10, 15};  
  print(arr,3);  
  return 0;  
}

**Basic Operation On One Dimensional Array**

**Function to traverse the array A**

void display(int A[], int n)  
{  
       cout<<"The elements of the array are:\n";  
       for(int i=0;i<n;i++)  
              cout<<A[i];  
}

**Function to Read elements of the array A**

void Input(int A[], int n)  
{   
      cout<<"Enter the elements:";  
      for(int i=0;i<n;i++)  
            cin>>A[i];  
}

**Function to Search for an element from A by Linear Search**

void lsearch(int A[], int n, int data)  
{  
      int i;   
      for(i=0; i<n; i++)  
      {  
            if(A[i]==data)  
            {  
                  cout<<"Data Found at : "<<i;  
                  return;  
            }  
      }  
      cout<<"Data Not Found in the array"<<endl;  
}

**Function to Search for an element from Array A by Binary Search**

int BsearchAsc(int A[], int n, int data)  
{  
       int Mid,Lbound=0,Ubound=n-1,Found=0;  
       while((Lbound<=Ubound) && !(Found))  
       {  
              Mid=(Lbound+Ubound)/2;        //Searching The Item  
              if(data>A[Mid])  
                     Lbound=Mid+1;  
              else if(data<A[Mid])  
                     Ubound=Mid-1;  
              else  
                     Found++;  
       }  
       if(Found)  
              return(Mid+1);        //returning 1ocation, if present  
       else  
              return(-1);        //returning -1,if not present  
}

**Function to Sort the array A by Bubble Sort**

void BSort(int A[], int n)  
{  
    int I,J,Temp;  
    for(I=0;I<n-1;I++) //sorting  
   {  
       for(J=0;J<(n-1-I);J++)  
            if(A[J]>A[J+1])  
           {  
               Temp=A[J]; //swapping  
               A[J]=A[J+1];  
               A[J+1]=Temp;  
           }  
    }  
}

**Function to Sort the array ARR by Insertion Sort**

void ISort(int A[], int n)  
{  
       int I,J,Temp;  
       for(I=1;I<n;I++) //sorting  
       {  
           Temp=A[I];  
           J=I-1;  
           while((Temp<A[J]) && (J>=0))  
           {  
               A[J+1]=A[J];  
               J--;  
           }  
           A[J+1]=Temp;  
       }  
}

**Function to Sort the array by Selection Sort**

void SSort(int A[], int n)  
{  
    int I,J,Temp,Small;  
    for(I=0;I<n-1;I++)  
    {  
         Small=I;  
         for(J=I+1;J<n;J++) //finding the smallest element  
         if(A[J]<A[Small])  
              Small=J;  
         if(Small!=I)  
        {  
            Temp=A[I]; //Swapping  
            A[I]=A[Small];  
            A[Small]=Temp;  
        }  
     }  
}

**Function to merge A and B arrays of lenghts N and M**

void Merge(int A[], int B[], int C[], int N, int M, int &K)  
{  
      int I=0, J=0;  
      K=0;             //Initialisation of counters for A, B, and C  
      while (I<N && J<M)  
      {  
            if (A[I]<B[J])  
                  C[K++]=A[I++];  
            else if (A[I]>B[J])  
                  C[K++]=B[J++];  
            else  
            {  
                  C[K++]=A[I++];  
                  J++;  
            }  
      }  
      for (int T=I;T<N;T++)  
            C[K++]=A[T];  
      for (T=J;T<M;T++)  
            C[K++]=B[T];  
}

**C-Strings (Character Arrays)**

**STRING:** It is an array of type char.

**Syntax for declaration**

char <array/string name> [max. number of characters to be stored +1];

The number of elements that can be stored in a string is always n-1, if the size of the array specified is n. This is because 1 byte is reserved for the NULL character '\0' i.e. backslash zero. A string is always terminated with the NULL character.

**Example:**  
char str[80];   
In the above example, str can be used to store a string with 79 characters.

**Initializing a string**

A string can be initialized to a constant value when it is declared.

char str[ ] = "Good";  
    Or  
char str[]={'G','o','o','d','\0'};

Here. 'G' will be stored in str[0], 'o' in str[1] and so on.

**Note:** When the value is assigned to the complete string at once, the computer automatically inserts the NULL character at the end of the string. But, if it is done character by character, then we have to insert it at the end of the string.

**Reading strings with/without embedded blanks**

To read a string without blanks cin can be used  
cin>>str;  
To read a string with blanks cin.getline() or gets() can be used.  
cin.getline(str,80);  
   -Or-  
gets(str);

**Printing strings**

cout and puts() can be used to print a string.  
cout<<str:  
   Or  
puts(str);

**Note:** For gets( ) and puts(), the header file <cstdio> (formally stdio.h) has to be included. puts() can be used to display only strings. It takes a line feed after printing the string.

|  |  |
| --- | --- |
| **cin** | **gets()** |
| It can be used to take input of a value of any data type. | It can be used to take input of a string. |
| It takes the white space i.e. a blank, a tab, or a new line character as a string terminator. | It does not take the white space i.e. a blank, a tab, or a new line character, as a string terminator. |
| It requires header file iostream.h | It requires the header file stdio.h |
| **Example:** char S[80]; cout << "Enter a string:”; cin>>S; | **Example:** char S[80]; cout << "Enter a string:"; gets(S); |

|  |  |
| --- | --- |
| **cout** | **puts()** |
| It can be used to display the value of any data type. | It can be used to display the value of a string. |
| It does not take a line feed after displaying the string. | It takes a line feed after displaying the string. |
| It requires the header file iostream.h | It requires the header file stdio.h |
| **Example:** char S[80] = "Computers"; cout<<S<<S;   **Output:** Computers Computers | **Example:** char S[80] = "Computers"; puts(S); puts(S);   **Output:** Computers Computers |

**Counting the number of characters in a string and printing it backwards**

#include<iostream>  
using namespace std;  
  
int main( )  
{  
  char str[80];  
  cout<<"Enter a string:";  
  cin.getline(str,80);  
  for(int l=0; str[l]!='\0';l++);  //Loop to find length  
    cout<<"The length of the string is : "<<l<<endl ;  
  for(int i=l-1;i>=0;i--)    //Loop to display the string backwards  
    cout<<str[i];  
  return 0;   
}

**Function to count the number of words in a string**

void count(char S[])  
{  
  int words=0;  
  for(int i=0;S[i]!='\0';i++)  
  {  
    if (S[i]==' ')  
      words++;                   //Checking for spaces  
  }  
  cout<<"The number of words="<<words+1<<endl;  
}

**Function to find the length of a string**

int length(char S[ ])  
{  
  for(int i=0;S[i]!='\0';i++);  
     return i;  
}

**Function to copy the contents of string S2 to S1**

void copy(char S1[ ], char S2[ ])  
{  
   for(int i=0;S2[i]!='\0';i++)  
      S1[i]=S2[i];  
   S1[i]='\0';  
}

**Function to concatenate the contents of string S2 to S1**

void concat(char S1[ ], char S2[ ])  
{  
   for(int l=0;S1[l]!='\0';l++);  
     for(int i=0;S2[i]!='\0';i++)  
         S1[l++]=S2[i];  
     S1[l]='\0';  
}

**Function to compare strings STR1 to STR2.**

The function returns a value>0 if //STR1>STR2, a value<0 if STR1<STR2, and value 0 if STR1=STR2

int compare(char STR1[ ],char STR2[])  
{  
  for(int I=0;STR1[I]==STR2[I] && STR1[I]!='\0'&&STR2[I]!='\0'; I++);  
     return STR1[I]-STR2[I];  
}

**To reverse the contents of string S and store it in string Rev**

void Reverse(char S[], char Rev[])  
{  
   for(int C1=0; S[C1]!='\0'; C1++);  
      C1--;  
   for(int C2=0;C1>=0;C2++,C1--)  
      Rev[C2]=S[C1];  
   Rev[C2]='\0';  
}

**Function to check whether a string S is a palindrome or not**

int Palin(char S[])  
{  
   for(int L=0;S[L]!='\0';L++);    //To find length  
      for(int C=0;(C<L/2) && (S[C]==S[L-C-1]);C++);  
         return (C==L/2)?1:0; //Returns 1 if Palindrome else 0  
}

**Function to change the case of string S to uppercase**

void Upper(char S[])  
{  
   for(int i=0;S[i]!='\0';i++)  
      S[i] = (S[i]>='a' && S[i]<='z')?(S[i]-32):S[i];  
}

**Function to change the case of string S to lower case**

void Lower(char S[])  
{  
   for(int i=0;S[i]!='\0';i++)  
       S[i] = (S[i]>='A' && S[i]<='Z')?(S[i]+32):S[i];  
}

**Function to extract n characters from left side of the string and store it in a different string.**

Example: 4 characters from ENVIRONMENT=ENVI

int SLeft(char S[ ], int n, char result[ ])  
{  
   for(int l=0;S[l]!='\0';l++);  
     if(n<=I)    //characters extracted should be <=length  
     {  
        for(int i=0;i<n;i++)  
            result[i]=S[i];  
        result[i]='\0';  
        return 1;  
     }  
     else  
        return 0;  
}

**Function to extract n characters from right side of the string and store it in a different string.**

Example: 4 characters from ENVIRONMENT=MENT

int SRight(char S[ ], int n, char result[ ])  
{  
   for(int l=0;S[l]!='\0';l++);  
     if(n<=I)     //characters extracted should be <=length  
     {  
        for(int j=0;i=l-n;S[i]!=’/0’;i++,j++)  
            result[j]=S[i];  
        result[j]='\0';  
        return 1;  
     }  
     else  
        return 0;  
}

**Structure [download](http://www.cppforschool.com/tutorial/pdf/chapter15-structure.pdf)**

A structure is a collection of variable which can be same or different types. You can refer to a structure as a single variable, and to its parts as **members** of that variable by using the dot (.) operator.  The power of structures lies in the fact that once defined, the structure name becomes a **user-defined data type** and may be used the same way as other built-in data types, such as int, double, char.

struct Student  
{  
 int rollno, age;

char name[80];

float marks;  
};

int main()  
{

// declare two variables of the new type

Student s1, s3;

//accessing of data members

cin >> s1.rollno >> s1.age >> s1.name >> s1.marks;

cout << s1.rollno << s1.age << s1.name << s1.marks;

   //initialization of structure variable

Student s2 = {100, 17, "Aniket", 92};

cout << s2.rollno << s2.age << s2.name << s2.marks;

  //structure variable in assignment statement

s3 = s2;

cout << s3.rollno << s3.age << s3.name << s3.marks;

return 0;

}

**Defining a structure**

When dealing with the students in a school, many variables of different types are needed.  It may be necessary to keep track of name,  age, Rollno, and marks point for example.

struct Student

{

int rollno, age;

char name[80];

float marks;

};

**Student**is called the **structure tag,**and is your brand new data type, like int, double or char.  
  
**rollno, name, age,**and **marks**are **structure members.**

**Declaring Variables of Type struct**

The most efficient method of dealing with structure variables is to define the structure **globally**.  This tells "the whole world", namely main and any functions in the program, that a new data type exists.  To declare a structure globally, place it **BEFORE** void main().  The structure variables can then be defined locally in main, for example…

struct Student

{

int rollno, age;

char name[80];

float marks;

};

int main()

{

// declare two variables of the new type

Student s1, s3;

………

………

return 0;

}

**Alternate method of declaring variables of type struct:**

struct Student

{

int rollno, age;

char name[80];

float marks;

} s1, s3;

**Accessing of data members**

The accessing of data members is done by using the following format:  
structure variable.member name  
for example

cin >> s1.rollno >> s1.age >> s1.name >> s1.marks;

**Initialization of structure variable**

Initialization is done at the time of declaration of a variable. For example

Student s2 = {100, 17, "Aniket", 92};

**Structure variable in assignment statement**

s3 = s2;

The statement assigns the value of each member of s2 to the corresponding member of s3. Note that one structure variable can be assigned to another only when they are of the same structure type, otherwise complier will give an error.

**Nested structure (Structure within structure)**

It is possible to use a structure to define another structure. This is called nesting of structure. Consider the following program

struct Day

{

    int month, date, year;

};

struct Student

{

int rollno, age;

char name[80];

Day date\_of\_birth;

float marks;

};

**Accessing Member variables of Student**

To access members of date\_of\_birth we can write the statements as below :

Student s; // Structure variable of Student

s.date\_of\_birth.month = 11;

s.date\_of\_birth.date = 5;

s.date\_of\_birth.year = 1999;

**typedef**

It is used to define new data type for an existing data type. It provides and alternative name for standard data type. It is used for self documenting the code by allowing descriptive name for the standard data type.  
  
The general format is:  
typedef existing datatype new datatype  
for example:  
typedef float real;  
Now, in a program one can use datatype real instead of float.  
Therefore, the following statement is valid:  
real amount;

**Enumerated data type**

The enum specifier defines the set of names which are stored internally as integer constant. The first name was given the integer value 0, the second value 1 and so on.

for example:

enum months{jan, feb, mar, apr, may} ;

It has the following features:

* It is user defined.
* It works if you know in advance a finite list of values that a data type can take.
* The list cannot be input by the user or output on the screen.

**#define preprocessor directive**

The #define preprocessor allows to define symbolic names and constants e.g.#define pi 3.14159   
This statement will translate every occurrence of PI in the program to 3.14159

**Macros**

Macros are built on the #define preprocessor. Normally a macro would look like:#define square(x) x\*x   
Its arguments substituted for replacement text, when the macro is expanded.