Logging in From a Linux System or Localhost
4 days 37 min ago
Auto response
4 days 6 hours ago
Re: final issue.
4 days 22 hours ago
Re: Rocky, thank you for this
4 days 23 hours ago

Learning C/C++ Step-By-Step - Page 07

Submitted by ganesh35 (Contact Author) (Forums) on Wed, 2009-01-07 18:29. ::

07. Step-by-Step C/C++ --- C Programming - Arrays

1. Introduction to arrays

A variable can hold a constant value. Only a single constant value and it is not possible to hold more than one value in a variable.

The following example demonstrates the scope a variable.

```
int main()
{
     int sno;
     sno = 1001;
     sno = 1008;
     sno = 1005;
     printf("%d", sno);
     return 0;
}
```

Output:
1005

The above program is able to display only 1005, but not all the values (i,e. 1001, 1008, 1005).

Can we substitute the following program in place of the above program.

```
int main()
{
     int sno;
     sno 0 = 1001;
     sno 1 = 1008;
     sno 2 = 1005;       printf("%d", sno);
     return 0;
}
```

Output:
Nothing,
The above program displays a list of errors, because of the approach is wrong.

Let's continue with the following program to get 0 errors program.

```
int main()
{
     int sno[3];
     sno[0] = 1001;
     sno[1] = 1008;
     sno[2] = 1005;
     printf("%d", sno[2] );
     return 0;
}
```

```
/* 3 values to be insert */
/* First location to insert 1001 */
/* Next location to insert 1008 */
/* and Next location to insert 1005 */
/* Prints the value of 2nd location */
```

Output:
Nothing

The above program displays a list of errors, because of the approach is wrong.

Depending on the above program, the variable **sno** can hold more than one student number. It's easy, by using multi-location technique, is also known as arrays.

2. About Arrays

Arrays contain a number of data items of the same type. This type can be a simple data type, a structure, or a class. The items in an array are called elements. Number accesses elements; this number is called an index. Elements can be initialized to specific values when the array is defined.

Arrays can have multiple dimensions.

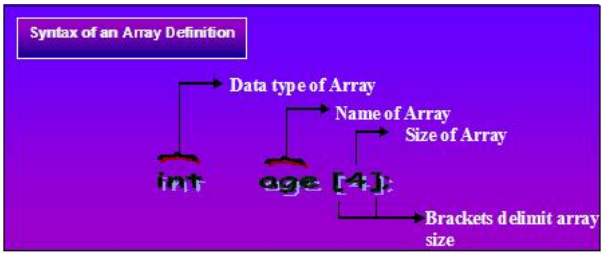
A two-dimensional array is an array of array. The address of an array can be used as an argument to a function; the array itself is not copied. Arrays can be used as member data in classes. Care must be taken to prevent data from being placed in memory outside an array.

/* The following program reads 4 persons age and displays it */

```
/* 47_arrays.c */
#include <stdio.h>
int main()
{
     int age[4], i;
     for( i=0; i<4; i++)
     {
          printf("Enter an age "); scanf("%d", &age[i]);
     }

     for(i=0; i<4; i++)
          printf("\nYou entered %d", age[i]);

     return 0;
}
```

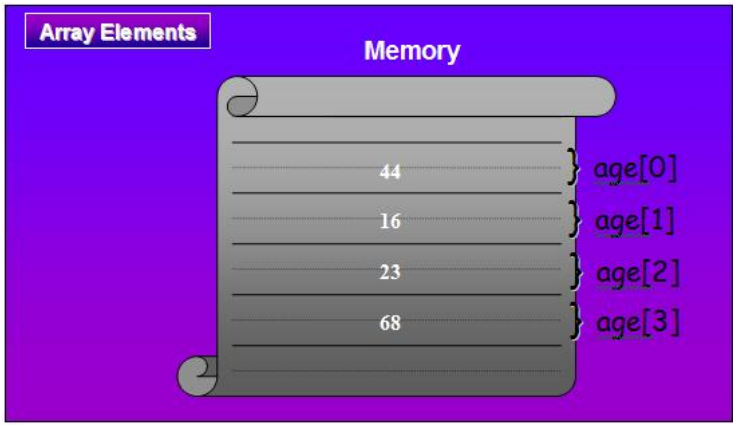


Like other variables in C, an array must be defined before it can be used to store information. And, like other definitions, an array definition specifies a variable type and a name.

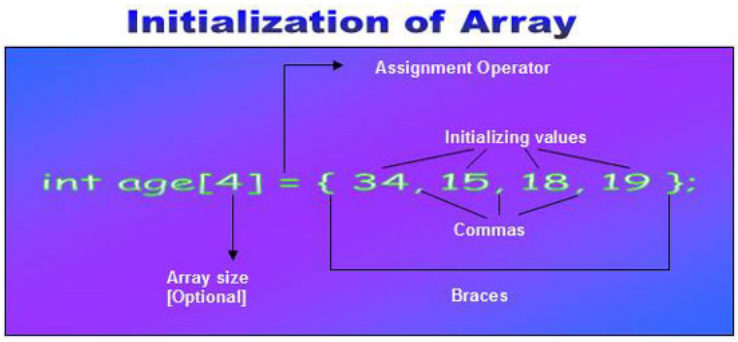
But it includes another feature: a **size**. The size specifies how many data items the array will contain. It immediately follows the name, and is surrounded by square brackets.

3. Array Elements

The items in an array are called elements. Single Dimensional array accepts values to either row wise or column wise. It can store only one set of values.
The first array element is 0, second is 1 and so on.



An array value can be initialized directly at design time.
Initialization of arrays is as follows..



4. Passing Arrays to Functions

Arrays can be used as arguments to functions.
In a function declaration, the data type and sizes of the array represent array arguments.

```
void display(float [DISPLAY][MONTHS]);
```

When the function is called, only the name of the array is used as an argument.

```
display(sales);
```

Program to accept and print array of 10 elements

```
/* 48_ ele10.c */
#define MAX 10
display(int a[MAX])
{
     int i;
     for(i = 0;i<MAX; i++)
          printf("\n%d",,a[i]);
}

int main()
{
     int x[MAX], i;
     for(i=0;i<MAX; i++)
          scanf("%d",&x[i]);

     display(x);
```

```
     return 0;
}
```

Returning array of values from functions is also possible but we must be clear with the concept of pointers. Please look in to the pointers topics for more info.

5. Classification of Arrays

Arrays are of two types.

1. Single dimensional Arrays
2. Multi Dimensional Arrays

1. Single Dimensional Arrays

A single dimensional array is a collection of elements in a row or a column fashion.
A single dimensional array can accept the following operations.

1. Append element
2. Insert element
3. Delete element
4. Replace element
5. Search element
6. Deletion of array
7. Sorting of an array

The following program is able to perform all the tasks described above.

```
/* ARRAY FUNCTIONS */
/* 49_sarray.c */
#define N 100
#define M 10
int i,j,r,c,r1,r2,c1,c2;
```

```
/* Read Array elements */
int accept_values(int a[N])
{
     int n;
     printf("\nHow many values you wish to enter..? ");
     scanf("%d",&n);
     printf("\nEnter the data elements..\n");
     for(i=0;i<n;i++)
          scanf("%d",&a[i]);
     return n;
}
```

```
/* Display array elements */
void display(int a[N],int n)
{
     printf("\n Array elements are...");
     for(i=0;i<n;i++)
          printf("\n%d",a[i]);
}
```

```
/* Delete array element */
int delete_cell(int a[N],int n)
{
     int pos;
     char ch;
     printf("\nEnter the position of element to be deleted: ");
     scanf("%d",&pos);
     for(i=pos-1;i<n;i++)
          a[i] = a[i+1];
     n--;
     printf("\n Do you wish to continue..(y/n)?");
     ch = getche();
     if(ch == 'y') delete_cell(a,n);
     return n;
}
```

```
/* Insert array element */
int insert_cell(int a[N],int n)
{
     int pos, new;
     char ch;
     printf("\nEnter the element to be inserted: "); scanf("%d",&new);
     printf("\nEnter the position of insertion: "); scanf("%d",&pos);
     for(i=n;i>=pos;i--)
          a[i] = a[i-1];
     a[pos-1] = new;
     n++;

     printf("\n Do you wish to continue..(y/n)?");
     ch = getche();
     if(ch=='y') insert_cell(a,n);
     return n;
}
```

```
/* To append element to an existing array */
int append_cell(int a[N],int n)
{
     int pos, new;
     char ch;
     printf("\nEnter the element to be appended: ");
     scanf("%d",&new);
     a[n] = new;
     n++;
     printf("\n Do you wish to continue..(y/n)?");
     ch = getche();
     if(ch=='y')append_cell(a,n);
     return n;
```

```
}
```

```
/* Sorting a list of elements of an array in descending order */
void sort_list_descend(int a[N],int n)
{
     int temp;
     for(i=0;i<n;i++)
          for(j=i;j<n;j++)
               if(a[i] < a[j])
               {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
               }
     printf("\nThe sorted elements in the descending order
}
```



```
/* Sorting a list of elements of an array in ascending or
void sort_list_ascend(int a[N],int n)
{
     int temp;
     for(i=0;i<n;i++)
          for(j=i;j<n;j++)
               if(a[i] > a[j])
               {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
               }
     printf("\nThe sorted elements in the ascending order is..");
}
```

```
/* To find the smallest and biggest of an existing array */
void small_big(int a[N],int n)
{
     int temp;
     for(i=0;i<n;i++)
          for(j=i;j<n;j++)
               if(a[i] < a[j])
               {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
               }
     printf("\nThe Smallest element is : %d",a[n-1]);
     printf("\nThe Biggest element is : %d",a[0]);
}
```

```
/* Search for an element in an array */
void search(int a[N],int n)
{
     int target, temp=0;
     printf("\n Enter element to be searched: ");
     scanf("%d",&target);
     for(i=0;i<n;i++)
          if(a[i] == target)
          {
               printf("\nFound at position no. %d",i+1);
               temp = 1;
          }
     if(temp == 0)
          printf("\n Not found");
}
```

```
/* Main program */ /* To demonstrate simple array operations */ #include<stdio.h>
#include<conio.h> #define m 100
int main()
{
     int a[m],n; char ch;
     clrscr();
     n = accept_values(a);
     do
     {
          printf("\n 1 - Append_cell");
          printf("\n 2 - Delete_cell");
          printf("\n 3 - Insert_cell");
          printf("\n 4 - Sort_list_descend");
          printf("\n 5 - Sort_list_ascend");
          printf("\n 6 - Small_big");
          printf("\n 7 - Search");
          printf("\n 8 - Remove_list");
          printf("\n 9 - Exit");
          printf("\n Enter your choice: ");          ch = getche();
          printf("\n");
          switch(ch)
          {
               case '1': n = append_cell(a,n); break;
               case '2': n = delete_cell(a,n); break;
               case '3': n = insert_cell(a,n); break;
               case '4': sort_list_descend(a,n); break;
               case '5': sort_list_ascend(a,n); break;
               case '6': small_big(a,n); break;
               case '7': search(a,n); break;
               case '8': n = 0; break;
               case '9': printf("\nThis will terminate your program."); break;
          }
          display(a,n);
          printf("\nDo you wish to run again..(y/n)?");
          ch = getche();
     }
     while(ch!='9');
     return 0;
}
```

2. Double Dimensional Arrays

A double dimensional array is a collection of elements in row and column fashion.
A Multi dimensional array can accept the following operations.

A multi dimensional array is commonly used in the areas of matrices to understand whole tasks in an easiest approach.

MATRIX FUNCTIONS

```
/* 50_menumat.c */
#define N 100
#define M 10
int i, j, r, c, r1, r2, c1, c2;
```

```
/* Read the values for a MATRIX */
void read_matrix(int A[M][M])
{
     printf("\nHow many rows? ");
     scanf("%d",&r);
     printf("\nHow many columns? ");
     scanf("%d",&c);
     for(i=0;i<r;i++)
          for(j=0;j<c;j++)
               scanf("%d",&A[i][j]);
}
```

```
/* Write the values of a MATRIX */
void disp_matrix(int A[M][M])
{
     for(i=0;i<r;i++)
     {
          for(j=0;j<c;j++)
          printf("%5d",A[i][j]);
          printf("\n");
     }
}
```

```
/* To find the TRANSPOSE for a MATRIX of ANY ORDER */
void tra_matrix_1(int T[M][M],int A[M][M])
{
     printf("\nTranspose of A is\n");
     for(i=0;i<c1;i++)
     {
          for(j=0;j<r1;j++)
          {
               T[i][j] = A[j][i];
               printf("%5d",T[i][j]);
          }
          printf("\n");
     }
}
```

```
/* To ADD two MATRICES ( possible,only if they are of EQUAL ORDER ) */
void add_matrix(int C[M][M],int A[M][M],int B[M][M])
{
     for(i=0;i<r;i++)
          for(j=0;j<c;j++)
               C[i][j] = A[i][j] + B[i][j];
     printf("\nSum of A and B is");
}
```

```
/* To MULTIPLY MATRICES of ANY ORDER ( provided they follow the MATRIX MULTIPLICATION RULE ) */
void mul_matrix_1(int C[M][M],int A[M][M],int B[M][M])
{
     int k;
     printf("\nProduct of A and B is..\n");
     printf("\nMatrix C\n");
     for(i=0;i<r1;i++)
     {
          for(j=0;j<c2;j++)
          {
               C[i][j]=0;
               for(k=0;k<r2;k++)
                    C[i][j] = C[i][j]+(A[i][k]*B[k][j]);
               printf("%5d",C[i][j]);
          }
           printf("\n");
      }
}
```

```
/* To SUBTRACT two MATRICES ( possible,only if they are of EQUAL ORDER ) */
void sub_matrix(int C[M][M],int A[M][M],int B[M][M])
{
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            C[i][j] = A[i][j] - B[i][j];
    printf("\nDifference of A and B is");
}
```

```
/* A MENU driven program to perform MATRIX operations */ #include <stdio.h>
#include <conio.h>
int main()
{
    int A[M][M],B[M][M],C[M][M],T[M][M]; char ch;
    clrscr( );
    printf("\nEnter matrix A elements..\n"); read_matrix(A); r1=r; c1=c;
    printf("\n Matrix A\n"); disp_matrix(A);
    printf("\nEnter matrix B elements..\n"); read_matrix(B); r2=r; c2=c;
    printf("\n Matrix B\n");disp_matrix(B);
    do {
        printf("\n1:Addition");
        printf("\n2:Subtraction");
        printf("\n3:Multiplication");
        printf("\n4:Transpose");
        printf("\n5:Exit");
        printf("\nEnter your choice.."); ch = getche();
        switch(ch)
```

```
        {
            case '1':
                    if(r1==r2 && c1==c2)
                    {
                        add_matrix(C,A,B);printf("\nMatrix C\n");
                        disp_matrix(C);
                    }
                    else
                    {
                        printf("\nYour entered values of r1,r2 && c1,c2 are not equal,");
                        printf("\nhence I cannot do this Matrix Addition.");
                        printf("\nPlease enter the correct matrices.");
                    } break;

            case '2':
                    if(r1==r2 && c1==c2)
                    {
                        sub_matrix(C,A,B);printf("\nMatrix C\n");
                        disp_matrix(C);
                    }
                    else
                    {
                        printf("\nYour entered values of r1,r2 && c1,c2 are not equal,");
                        printf("\nhence I cannot do this Matrix Subtraction.");
                        printf("\nPlease enter the correct matrices.");
                    } break;

            case '3':
                    if(c1==r2)
                        mul_matrix_1(C,A,B);
                    else
                    {
                        printf("\nColumns(c1) of Matrix A are NOT EQUAL TO");
                        printf(" Rows(r2) of Matrix B.");
                        printf("\nHence I cannot do this Matrix Multiplication.");
                        printf("\nPlease enter matrices such that c1 == r2.");
                    } break;

            case '4':   printf("\nOrder of Matrix A is %d x %d",r1,c1);
                        tra_matrix_1(T,A);
                        printf("\nOrder of A transpose is %d x %d",c1,r1); break;

            case '5': printf("\nThis will terminate your program."); break;
        }
        printf("\nDo you wish to run again...[y/n]?"); ch=getche();
    }while(ch!='5');
    return 0;
}
```

previous | up | next
Learning C/C++ Step-By-Step - Page 06 | Learning C/C++ Step-By-Step - Page 08

add comment | view as pdf | print: this | all page(s)

Related Tutorials

- Beginner's Guide To c++
- An Explanation of Pointers (C++)