Lecture 9 : Array

Acknowledgment : Lecture notes from Ohio Supercomputing Center

Warming Up Problem

- Problem : Get 5 integer numbers using scanf function, and print the numbers in reverse order.

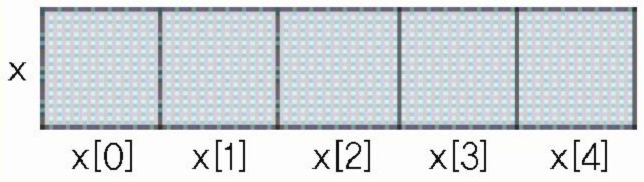
- Example
  - Input   : 3 10 5 7 2
  - Output : 2 7 5 10 3

```c
#include <stdio.h>

int main()
{
    int x0, x1, x2, x3, x4;

    printf("Get 5 integers : ");
    scanf("%d", &x0);
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &x3);
    scanf("%d", &x4);

    printf("Reverse Order : ");
    printf("%d ", x4);
    printf("%d ", x3);
    printf("%d ", x2);
    printf("%d ", x1);
    printf("%d ", x0);
    printf("\n");

    return 0;
}
```

$\rightarrow$

```c
#include <stdio.h>

int main()
{
    int x[5], i;

    printf("Get 5 integers : ");
    for (i = 0; i < 5; ++i)
        scanf("%d", &x[i]);

    printf("Reverse order : ");
    for (i = 4; i >= 0; --i)
        printf("%d ", x[i]);
    printf("\n");

    return 0;
}
```

Using array



Memory structure

Array Variables?

- Array variable holds a sequence of multiple values with same data type
  - int a[10];

- There is an ordered method for extracting individual data items
  - a[3]=7;

- Example
  - If we want to store scores for exam1 , exam2 , exam3 , ... , exam10
  - First try,

```
int exam1 = 90;   int exam2 =85;  …     int exam100=93;
sum = exam1 + exam2 + … + exam100;
```

  - Using array

```
int exam[100] = {90 , 85, … , 93};

for (i=0;i<100;i++) sum+=exam[i];
```

Array Variables Example

- **In C, array index starts at zero.**

```
int id[3]; /* declaration of array id,
            which has space for three consecutive int variables */

id[0] = 101;
id[1] = 232;
id[2] = 231;
```

- Each piece of data in an array is called an element. Thus, array id has three elements.

Array Declaration, Initialization

- Array Declaration
  - ***type arrayName[arraySize];***

```
float a[5];      // declare a float type array with size 5 and
int x, y, b[5];
```

- Array Initialization

```
int a[5] = {1, 3, 5, 7, 9};
int a[] = {1, 3, 5, 7, 9};

a[2] = 11;     // a -> {1, 3, 11, 7, 9}
a[5] = 20;     // wrong!!!!    Index can be between 0 and 4
```

example

```c
#include <stdio.h>

int main()
{
    int x[] = {1, 3, 5, 7, 9};

    printf("           x = %8u\n", x);
    printf("   sizeof(x) = %8d\n", sizeof x);
    printf("sizeof(x[0]) = %8d\n", sizeof x[0]);

    return 0;
}
```

output

```
           x =  2280640
   sizeof(x) =       20
sizeof(x[0]) =        4
```

Multi-Dimensional array

- Multi-dimensional arrays have two or more index values which are used to specify a particular element in the array. For this 2D array element,

```
image[i][j]
```

- the first index value **i specifies a row index, while j specifies a column index.** Declaring multi-dimensional arrays is similar to the 1D case:

```
int a[10]; /* declare 1D array */
float b[3][5]; /* declare 2D array */
double c[6][4][2]; /* declare 3D array */
```

- Note that it is quite easy to allocate a large chunk of **consecutive memory** with multi-dimensional arrays. Array **c contains 6x4x2=48 doubles.**

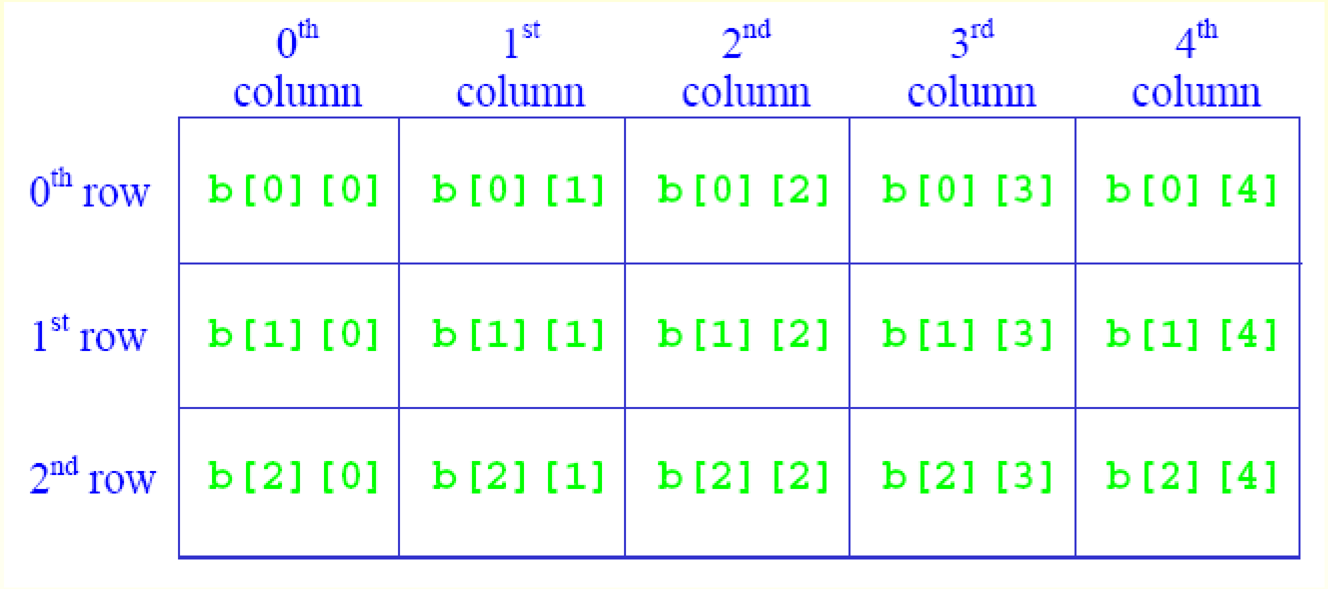
- Initialization

```
int age[2][3] = { {4,8,12} , {19,6,-1} };
```

Multi-Dimensional array

- Very commonly used for matrix

```
int b[3][5];
for (i=0;i<3;i++) {
    for (j=0;j<5;j++) sum+=b[i][j];
}
```



	0th column	1st column	2nd column	3rd column	4th column
0th row	b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[0][4]
1st row	b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[1][4]
2nd row	b[2][0]	b[2][1]	b[2][2]	b[2][3]	b[2][4]

Passing Array Argument

- Pass **address** of an array into function
- Example

```
int getSum (int score[] , int arraySize)     // int getSum (int* score , int arraySize)
{
    int i;
    int sum=0;
    for (i=0;i<arraySize;i++) sum+=score[i];
    return sum;
}

int main()
{
    int a[]={1,5,10};
    int sum=0;
    printf("a[0]=%d, a[1]=%d, a[2]=%d , sum=%d\n",a[0],a[1],a[2],sum);
    sum = getSum(a,3);
    printf("a[0]=%d, a[1]=%d, a[2]=%d , sum=%d\n",a[0],a[1],a[2],sum);
    return 0;
}
```

Array Arguments

- Because we pass address of array to function, array element value which is modified in a function will remain modified even after the function call is finished.
- example

```
int getSum (int score[] , int arraySize)
{
          int i;
          int sum=0;
          for (i=0;i<arraySize;i++) {
                    sum+=score[i];
                    score[i]=sum;
          }
          return sum;
}

int main()
{
          int a[]={1,5,10};
          int sum=0;
          printf("a[0]=%d, a[1]=%d, a[2]=%d , sum=%d\n",a[0],a[1],a[2],sum);
          sum = getSum(a,3);
          printf("a[0]=%d, a[1]=%d, a[2]=%d , sum=%d\n",a[0],a[1],a[2],sum);
          return 0;
}
```

Example : rcSum.c

```c
#include <stdio.h>
#include <string.h>     // strlen

#define N 3

void readNxN(int a[N][N]);
void sumNxN(const int a[N][N], int rSum[N], int cSum[N]);
void printNxN(const int a[N][N], const int rSum[N], const int cSum[N]);

int main()
{
    int x[N][N];
    int rSum[N] = {0}, cSum[N] = {0};

    readNxN(x);
    sumNxN(x, rSum, cSum);
    printNxN(x, rSum, cSum);

    return 0;
}
```

```
void readNxN(int a[N][N])
{
    int i, j;
    printf("Input %d x %d integer matrix :\n", N, N);
    for (i = 0; i < N; ++i)
        for (j = 0; j < N; ++j)
            scanf("%d", & a[i][j]);
}

void sumNxN(const int a[N][N], int rSum[N], int cSum[N])
{
    int i, j;
    for (i = 0; i < N; ++i) {
        for (j = 0; j < N; ++j) {
            rSum[i] += a[i][j];
            cSum[j] += a[i][j];
        }
    }
}
```

```
void printNxN(const int a[N][N], const int rSum[N], const int cSum[N])
{
    int i, j;
    const char *lseg = "-------";
    const int width = strlen(lseg) - 1;

    for (i = 0; i < 3; ++i) {
        for (j = 0; j < 3; ++j)
            printf("%*d ", width, a[i][j]);
        printf("| %*d\n", width, rSum[i]);
    }
    for (i = 0; i < 3; ++i)
        printf("%s", lseg);
    printf("+\n");
    for (i = 0; i < 3; ++i)
        printf("%*d ", width, cSum[i]);
    printf("\n");
}
```