

ARRAYS and STRUCTURES

Operating on specific values of a 1D array

Here is a program to generate an array B[40] containing values from 1 to 40, and also to find the sum of the first and last element of the array.

```
#include <stdio.h>
int main(void)
{
    int i,sum;
    int B[40];
    //generate the array
    for (i= 0 ; i < 40      ; i++)
        B[i] = i+1;
    sum = B[0] + B[39];
    printf("sum = %d", sum);
    printf("\n");
    return 1;
}
```

sum = 41

Updating specific elements of an array by a function

If specific elements of an array are to be updated by a function, the elements have to be passed-by-reference to the function, just like individual variables are sent to a function for getting updated. In the following the first and second elements of an array are being sent to a function which sends them after adding 10 to each value.

```
#include <stdio.h>
int main(void)
{
    int i,sum;
    int B[10];
    for (i= 0 ; i < 10      ; i++)
        B[i] = i+1;
    modify( &B[0], &B[1] )
    for (i= 0 ; i < 10      ; i++)
        printf(" %d", B[i]);
    return 1;
}
```

```

void modify ( int *a, int *b)
{
    *a = *a + 10;
    *b = *b + 10;
}

```

11 12 3 4 5 6 7 8 9 10

Printing an 1-dimensional array on multiple lines

Suppose the array B contains 40 '*' (stars), and we want to print 8 stars on each line. We would have 5 lines, so the variable i would go from 0 to 4. Each line will be a multiple of 8 plus the jth element, that means we need to print the (8i+j)th element of the array on the ith line. Here is the program :

```

#include <stdio.h>

int main(void)
{
    int i,j;
    char B[40];
    for (i= 0 ; i < 40 ; i++)
        B[i] = '*';
    for (i= 0 ; i < 5 ; i++)
    {
        for (j= 0 ; j < 8 ; j++)
            printf(" %c", B[8*i+j]);
        printf("\n");
    }
    return 1;
}

```

To print 10 stars in each line the program would need to be modified as follows:

```

#include <stdio.h>

int main(void)
{
    int i,j;
    char B[40];
    for (i= 0 ; i < 40 ; i++)
        B[i] = '*';
    for (i= 0 ; i < 4 ; i++)

```

```

        {
            for (j= 0 ; j < 10      ; j++)
                printf(" %c", B[10*i + j]);
            printf("\n");
        }
    return 1;
}

```

To convert an 1-dimensional array into a 2-dimensional array

Suppose we have an 1D array containing 40 integers and we want to convert it into a 2D array of size 5x8, we can proceed in the manner indicated in the above program.

```

#include <stdio.h>

int main(void)
{
    int i,j;
    char B[40], BB[5][8];
    for (i= 0 ; i < 40      ; i++)
        B[i] = '*';
    for (i= 0 ; i < 5      ; i++)
        for (j= 0 ; j < 8      ; j++)
            BB[i][j]= B[8*i+j];

    for (i= 0; i < 5 ; i++)
    {
        for (j= 0 ; j < 8; j++)
            printf(" %c", BB[i][j]);
        printf("\n");
    }
    return 1;
}

```

To print a specific row of a 2D array

Suppose we want to print the first row of a 2D array, we can fix the value of *i* and simply have a for loop to print each element using variable *j*.

```
#include <stdio.h>

int main(void)
{
    int i,j;
    int B[40], BB[5][8];
    for (i= 0 ; i < 40      ; i++)
        B[i] = i;
    for (i= 0 ; i < 5      ; i++)
        for (j= 0 ; j < 8      ; j++)
            BB[i][j]= B[8*i+j];

    i= 0;
    for (j= 0 ; j < 8; j++)
        printf(" %c", BB[i][j]);
    printf("\n");

    return 1;
}
```

Structures

While an array can hold items of same type, a structure will permit you to combine items of different type. Suppose you wanted to store names of 100 students and grades obtained by them in two exams. You could one array `NAME[100]` to store all the names, another array `GRADE1[100]` to store grades obtained in exam1 and finally one more array `GRADE2[100]` to store the grades of exam2.

NAME[100]

James
Hillary
Joan
Anil
Chen
Mendez
.....

GRADE1[100]

70
85
56
43
92
80
.....

GRADE2[100]

45
56
67
78
89
12
.....

To process the data using functions, you would have to pass the 3 separate arrays. However, you can use a structure to hold items of different types together. If you declare a structure to hold name, grade1 and grade2, as shown below

```
struct class {
    char name[20],
    int grade1;
    int grade2;
};
```

then you can declare an array of structures to hold all the items together as one single entity as shown below:

```
struct class CLASS[100];
```

CLASS[100]

James	70	45
Hillary	85	56
Joan	56	67
Anil	43	78
Chen	92	89
Mendez	80	12
.....

Individual items from this structure can be obtained as follows.

```
CLASS[i].name  
CLASS[i].grade1  
CLASS[i].grade2
```

Consider another example from a retail outlet, where one wants to store the name of a product, quantity in stock and its price as one single structure called item.

```
struct item {  
    char name[20],  
    int quantity;  
    double price;  
};
```

then to store the information pertaining to 50 products, we can declare an array of structures as follows:

```
struct item ITEMS[50];
```

To get the quantity of the first item we shall use
ITEMS[0].quantity

and to get the price of the last item we shall use
ITEMS[49].price

and to get the name of the 10th item we shall use
`ITEMS[9].name`

Scrabble using structs

In this example, we will have a struct that stores a single tile from the game of Scrabble. We will declare an array of these structs to represent the tiles on one person's tray. We will simply read these in and then print them out, along with their aggregate score. In this example, pay attention to the use of arrays of structs and accessing components of the individual structs.

```
#include <stdio.h>

struct tile {
    char letter[10];
    int score;
};

void printTiles(struct tile mytiles[], int n);
void printScore(struct tile mytiles[], int n);

int main() {

    int i;
    //declare an array of tile structure
    struct tile mytiles[7];
    // Read in all tiles from user.
    for (i=0; i<7; i++) {
        scanf("%s", mytiles[i].letter);
        scanf("%d", &mytiles[i].score);
    }
    printTiles(mytiles, 7);
    printScore(mytiles, 7);
    return 0;
}

void printTiles(struct tile mytiles[], int n) {

    int i;
    for (i=0; i<n; i++)
        printf("%s ", mytiles[i].letter);
    printf("\n");
}

void printScore(struct tile mytiles[], int n) {
```

```

int i, sum=0;
for (i=0; i<n; i++)
    sum = sum + mytiles[i].score;
printf("Score = %d\n", sum);
}

```

Passing names and grades to functions using structs

We now consider an example where we have a structure for a student in a class. The structure will contain the name of the student and grade obtained in one exam. Then we use this structure to store information for a section of cop3223 having 200 students. We use two functions, one to print the name and grade in an organized way, and the other to print the average grade for the whole class.

```

#include <stdio.h>

struct class {
    char name[20];
    int grade;
};

void printClasss(struct class cop3223[], int n);
void printAverage(struct class cop3223[], int n);

int main() {

    int i;
    //declare an array of class structure
    struct class cop3223[200];

    // Read in all classs from user.
    for (i=0; i<200; i++) {
        printf("\nenter name and grade for student %d",i+1);
        scanf("%s", &cop3223[i].name);
        scanf("%d", &cop3223[i].grade);
    }

    printClasss(cop3223, 200);
    printAverage(cop3223, 200);
    return 0;
}

```

```

void printClasss(struct class cop3223[], int n) {

    int i;
    for (i=0; i<n; i++)
        printf("%s \t %d \n", cop3223[i].name,
                cop3223[i].grade);
    printf("\n");
}

void printAverage(struct class cop3223[], int n) {

    int i, sum=0;
    double av;
    for (i=0; i<n; i++)
        sum = sum + cop3223[i].grade;
    av = sum *1.0/n;
    printf("average Grade = %.2f\n", av);
}

```

Passing grades to functions using different names for structs

We now consider an example where we have a structure containing two grades for each student and we need to store the average grade for each student in the same structure. In this example we shall also show that we can use different name for the array of structures inside the function, and how the function updates the values in the main.

```

#include <stdio.h>

struct class {
    char name[20];
    int grade1;
    int grade2;
    double av;
};

void computeAverage(struct class dd[], int n);

int main() {

    int i;

```

```

struct class cop3223[200];

// Read in all classes from user.
for (i=0; i<200; i++) {
    printf("\nEnter name and grades for student %d",i+1);
    scanf("%s", &cop3223[i].name);
    scanf("%d %d", &cop3223[i].grade1, &cop3223[i].grade2);
}

computeAverage(cop3223, 200);

for (i=0; i< 200; i++)
    printf("%s \t %d \t %d \t %.2f\n", cop3223[i].name,
        cop3223[i].grade1, cop3223[i].grade2,cop3223[i].av);
return 0;
}

void computeAverage(struct class dd[], int n) {

    int i, sum=0;
    double av;
    for (i=0; i<n; i++)
        dd[i].av = (dd[i].grade1 + dd[i].grade2)/2.0;
}

```