Mahidol University

ITCS113 Fundamentals of Programming

Lecture 6 - 1D Array

Instructor: Asst. Prof. Dr. Akara Supratak

Contact: akara.sup@mahidol.edu

Wisdom of the Land

Agenda

- 1D Array
- Loop through 1D array

Mahidol University

1D Array

1D Array

Array Declaration (cont.)

- Number of the elements in the array can be a constant whole number

```
#define N 5
int grades[N];
```

Use #define to create a constant for an array size (at compile-time)

```
int n=5;
int grades[n];
```

Use a variable to specify the size of an array (at compile-time)

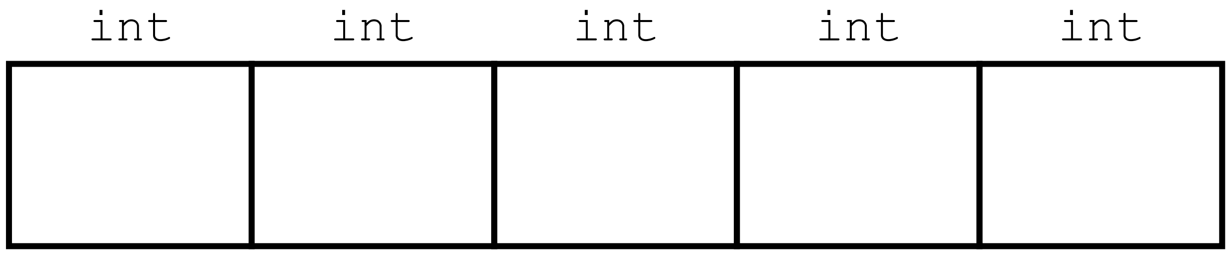
```
int n;
scanf("%d", &n);
int grades[n];
```

Use a variable to specify the size of an array (at run-time)

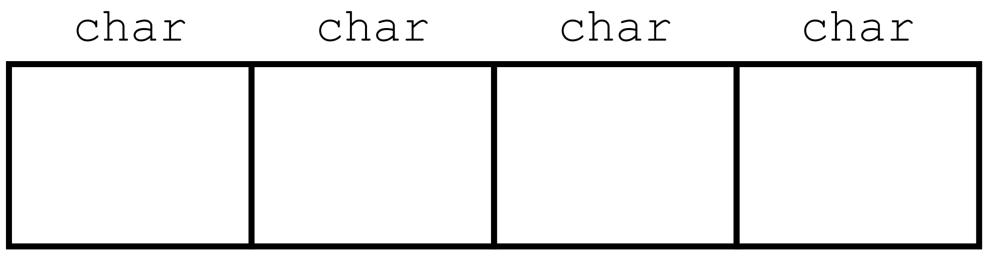
Examples

- To create an array, we use a declaration statement

```
int grades[5];
```



```
char codes[4];
```



**** Number of the element (`N` or `n`) must be declared and initialized (or assigned value) **before** being used in an array declaration ****

```
#define N 5
int grades[N];
```

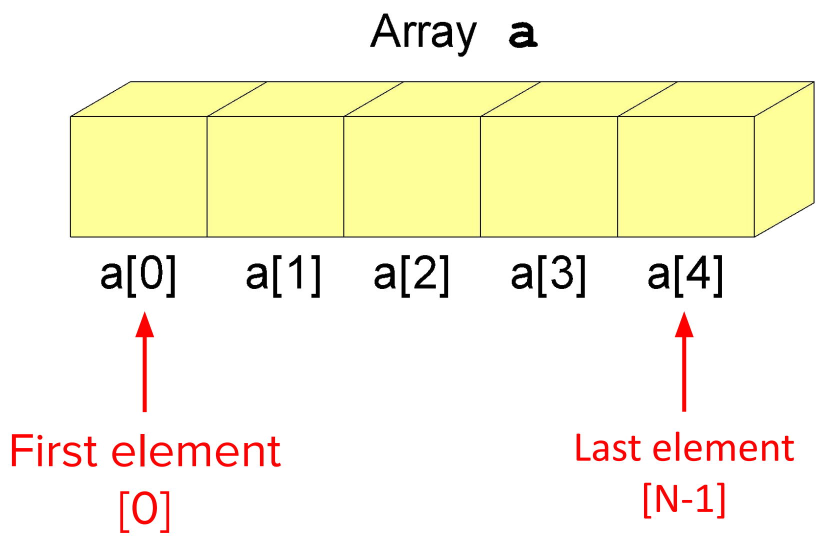
```
int n=5;
int grades[n];
```

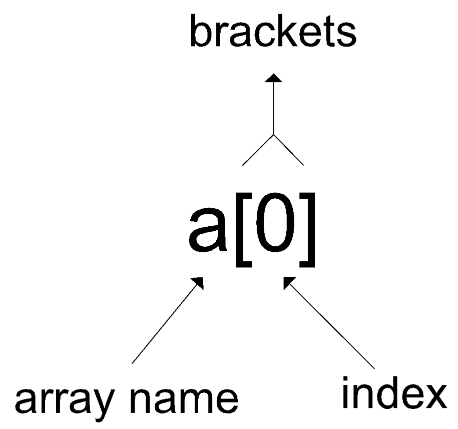
```
int n;
scanf("%d", &n);
int grades[n];
```

1D Array - Indexing

Indexing

- Any element can be accessed by giving the name of the array and the element's index (or position)

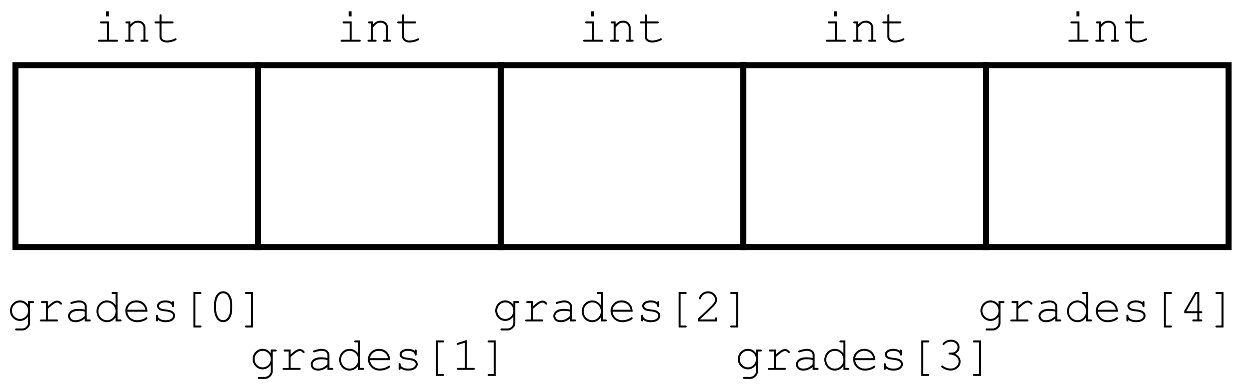




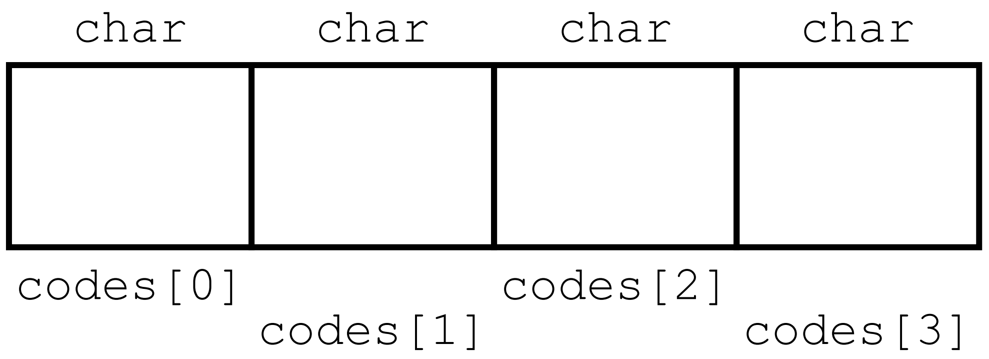
Examples

- Identifying individual array elements

```
int grades[5];
```



```
char codes[4];
```



1D Array - Initialization

Array Initialization

- In `C`, the initial values of elements in arrays are undefined
- We can assign values to array elements either by going through each element in the array or initializing within the declaration statement

```
int grades[5];
grades[0] = 98;
grades[1] = 87;
grades[2] = 92;
grades[3] = 79;
grades[4] = 85;
```

```
int grades[5] = {98, 87, 92, 79, 85};
```

Initializing within the Declaration

```
// Initialize within the declaration
// statement
int grades[5] = {98, 87, 92, 79, 85};
char codes[4] = {'x', 'a', 'm', 'n'};


// Initialize without specifying the size
int grades[] = {98, 87, 92, 79, 85};
char codes[] = {'x', 'a', 'm', 'n'};
```

Initializing within the Declaration

- If you **partially** initialize an array, the compiler sets the remaining elements to zero

```
float length[7] = {8.8, 6.4, 4.9, 11.2};
printf("%.2f\n", length[1]);      printf("%.2f\n", length[3]);

printf("%.2f\n", length[5]); //0.00

char codes[6] = {'x', 'a', 'm', 'n'};
printf("%c\n", codes[5]); //Check ASCII Tab :)
printf("%d\n", codes[5]); //0
```

- Thus, it's easy to initialize all the elements of an array to zero as follows:

```
float length[7] = {0};
```

Array Declaration vs. Initialization

```
#include <stdio.h>
#define N 5

int main() {
    int n1 = 5;
    int n2;
    scanf("%d", &n2);

    int array1[n1];           // OK, if no initialization
    int array2[n2];           // OK, if no initialization
    int array3[] = {0};       // OK, if the size is not specified
    int array4[N] = {0};      // OK, if constant is used
    int array5[n1] = {0};     /* Error: var.-sized object may not
                                 be initialized  */
    int array6[n2] = {0};     /* Error: var.-sized object may not
           be initialized  */
    return 0;
}
```

Mahidol University

Loop Through 1D Array

1D Array - Loop

Using loops for manipulating arrays

- We can use **any** expression of type **`int`** as an array index, e.g. `a[i]`, `a[i+1]`, etc.
- We can run the same code block for each element of an array.

```
int zeros[10];
zeros[0] = 0;
zeros[1] = 0;
zeros[2] = 0;
zeros[3] = 0;
zeros[4] = 0;
...
zeros[9] = 0;
```

⟷

```
int zeros[10];
for (int i=0; i<10; i++) {
    zeros[i] = 0;
}
```

Exercise

What is the output of the following program?

```
/* # 1 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 0; i < 5; i++) {
        printf("%d %d\n", i, arr[i]);
    }
    return 0;
}
```

Exercise

What is the output of the following program?

```
/* # 1 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 0; i < 5; i++) {
        printf("%d %d\n", i, arr[i]);
    }
    return 0;
}
```

```
0 2
1 4
2 6
3 0
4 1
```

Exercise

What is the output of the following program?

```
/* # 2 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 1; i <= 5; i++) {
        printf("%d %d\n", i, arr[5-i]);
    }
    return 0;
}
```

Exercise

What is the output of the following program?

```c
/* # 2 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 1; i <= 5; i++) {
        printf("%d %d\n", i, arr[5-i]);
    }
    return 0;
}
```

```
1 1
2 0
3 6
4 4
5 2
```

Exercise

What is the output of the following program?

```
/* # 3 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 0; i <= 5; i++) {
        printf("%d %d\n", i, arr[5-i]);
    }
    return 0;
}
```

Exercise

What is the output of the following program?

```
/* # 3 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    for (int i = 0; i <= 5; i++) {
        printf("%d %d\n", i, arr[5-i]);
    }
    return 0;
}
```

Don’t know the output :(because `arr[5]` is outside the boundary of the `arr` array, i.e., garbage value

Exercise

What is the output of the following program?

```
/* # 4 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    int o = -1;
    for (int i = 0; i < 5; i++) {
        if (arr[i] == 0) {
            o = i;
        }
    }
    printf("%d\n", o);
    return 0;
}
```

Exercise

What is the purpose of the following program?
What is the purpose of the program?

```
/* # 4 */
int main()
{
    int arr[5] = {2, 4, 6, 0, 1};
    int o = -1;
    for (int i = 0; i < 5; i++) {
        if (arr[i] == 0) {
            o = i;
        }
    }
    printf("%d\n", o);
    return 0;
}
```

```
3
```

Exercise

What is the output of the following program?

```
/* # 5 */
#include <stdio.h>
#define N 3
int main(){
    float prices[N] = {1.5, 3.5, 2.0};
    int units[N] = {4, 2, 3};
    float sales = 0;

    for(int i=0;i<N;i++){
        sales += prices[i]*units[i];
    }
    printf("%.2f", sales);
}
```

Exercise

What is the output of the following program?
What is the purpose of the program?

```c
/* # 5 */
#include <stdio.h>
#define N 3
int main(){
    float prices[N] = {1.5, 3.5, 2.0};
    int units[N] = {4, 2, 3};
    float sales = 0;

    for(int i=0;i<N;i++){
        sales += prices[i]*units[i];
    }
    printf("%.2f", sales);
}
```

```
19.00
```

Exercise

What is the output of the following program?

```
/* # 6 */
#include <stdio.h>
#define N 5
int main(){
    char alp[N] = {'A','B','C','D','E'};
    int  num[N] = {1,2,3,4,5};

    for(int i=0; i<N; i++){
        for(int j=N-1; j>=0; j--){
            printf("%c%d ", alp[i], num[j]);
        }
        printf("\n");
    }
    return 0;
}
```

Exercise

What is the output of the following program?

```c
/* # 6 */
#include <stdio.h>
#define N 5
int main(){
    char alp[N] = {'A','B','C','D','E'};
    int  num[N] = {1,2,3,4,5};

    for(int i=0; i<N; i++){
        for(int j=N-1; j>=0; j--){
            printf("%c%d ", alp[i], num[j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
A5 A4 A3 A2 A1
B5 B4 B3 B2 B1
C5 C4 C3 C2 C1
D5 D4 D3 D2 D1
E5 E4 E3 E2 E1
```

Tip - Array Declaration

- If the size is fixed, declare the SIZE of an array as a **constant**, e.g., **`#define N 5`**

```
#define N 5
...
int  array_num[N];
```

- If the size is specified by a user, receive an integer from a users and declare the array

```
int n;
scanf("%d", &n);
int array_num[n];
```

Tip - Array Initialization

- If all values in the array are known, you are allowed to initialized as assignment

```
#define N 3
...
int array_num[N]={1,2,3};
int array_num2[]={1,2,3};
```

- If the values are specified by a user, use a loop to receive all values

```
int array_num[n]; //n=5
for (int i = 0; i < 5; i++) {
 scanf("%d",&array_num[i]);
}
```

Mahidol University

Lab Exercises