Basis and Practice in Programming

Chapter 8: Arrays

Prof. Tamer ABUHMED
College of Software

성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

Lecture Objectives

- break & continue Statements
- Introduced goto statements
- Introduced the single dimension arrays in C language
- Introduced the rand() and srand() functions
- Introduced the multidimension arrays in C language
- Introduced the variable size arrays in C language
- Keywords

The break & continue Statements

- Find a secret number using a while loop
  - Note that “while(1)” is an infinite loop

```
# include <stdio.h>
# define SECRET 8

void main()
{
    int num;

    printf("Enter the secret number: ");
    while(1)    // infinite loop
    {
        scanf(“%d”, &num);
        if(num == SECRET){
            printf(“CORRECT! Congratulations.\n”);
            break;  // end the while loop
        }else
        {
            printf(“Try again: “);
            continue; // go to next iteration
        }
    }
}
```

Output

```
Enter the secret number: 5
Try again: 2
Try again: 4
Try again: 8
CORRECT! Congratulations.
```

goto

- Don’t use the goto statement ☺
  - But here how it works!

```
int main(void) {

    int salary = 15000;
    int age = 65;
    if (age > 60)
    {
        salary *= 1.5;
        goto a;
    }
    else {
        salary *= 1.2;
        goto b;
    }

a: salary += 100;
b: salary += 97;
    printf("Salary = %d", salary);
    return 0;
}
```

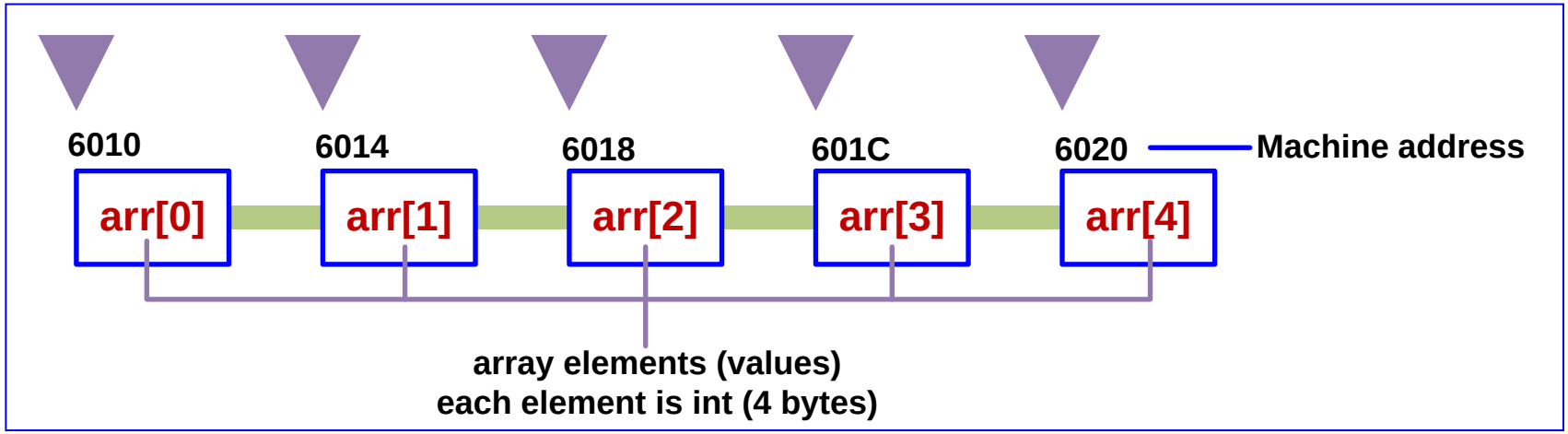
Output

```
Salary = 22697
```

Arrays

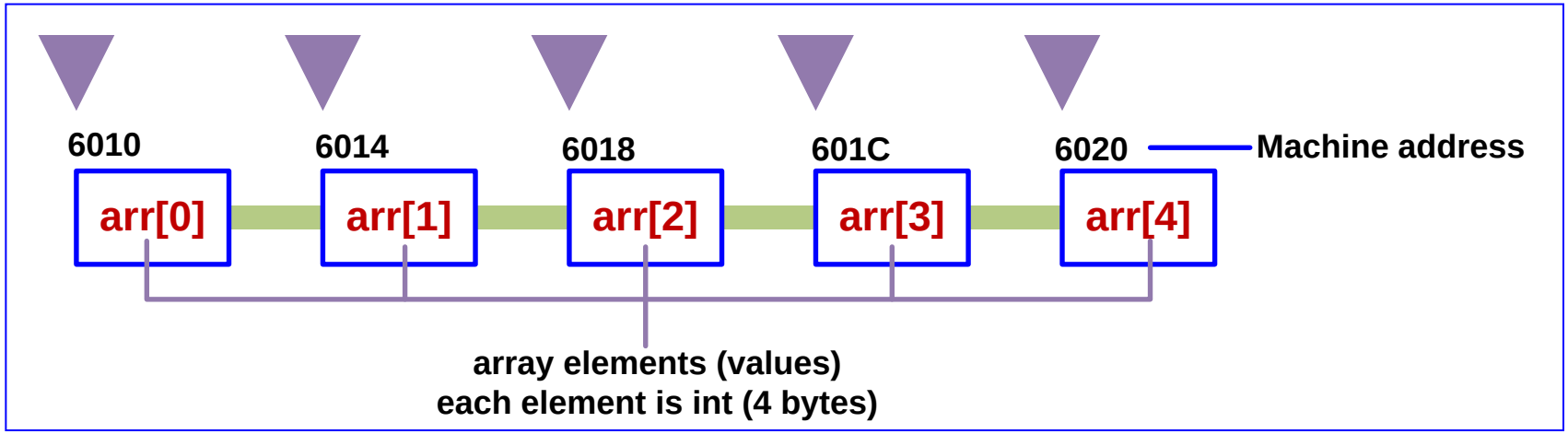
- An array
  - A sequence of elements of the same data type

  - **Example**
    - int arr[5]; // an array of 5 integer elements
    - Therefore, arr[0] ~ arr[4] are integers each stored in a 4 bytes memory block
    - The array name is the address of the first element of the array



Arrays – contd.

- An array – contd.
  - The array name is the address of the first element of the array
    - printf("The address of the array is %p\n", arr);
      - Prints the address of the first byte of the first element (6010 in the given example)
    - printf("The address of the array is %p\n", &arr[0]);
      - The first element of array "arr", arr[0], is an integer
      - So &arr[0] is the address of the first byte of the first element in the array "arr"



Arrays – contd.

- An example

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

void main()
{
     int arr_i[N], ii = 0; // int array of size N
     double arr_d[N];      // double array of size N

     for(ii = 0; ii < N; ii++){
          arr_i[ii] = N - ii;

          // print the integer array
          printf("arr_i[%d] = %d and saved at location %p\n",
                         ii, arr_i[ii], &arr_i[ii]);
     }

     for(ii = 0; ii < N; ii++){
          arr_d[ii] = (double)(N - ii) / N;

          // print the double array
          printf("arr_d[%d] = %.1lf and saved at location %p\n",
                    ii, arr_d[ii], &arr_d[ii]);
     }

}
```

Output

```
arr_i[0] = 5 saved at location 0012FF6C
arr_i[1] = 4 saved at location 0012FF70
arr_i[2] = 3 saved at location 0012FF74
arr_i[3] = 2 saved at location 0012FF78
arr_i[4] = 1 saved at location 0012FF7C

arr_d[0] = 1.0 saved at location 0012FF40
arr_d[1] = 0.8 saved at location 0012FF48
arr_d[2] = 0.6 saved at location 0012FF50
arr_d[3] = 0.4 saved at location 0012FF58
arr_d[4] = 0.2 saved at location 0012FF60
```

Arrays – contd.

- What is new?
  - %p: indicates an address (or pointer)

- More examples on arrays

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

void main()
{
     float candy[365];            /* array of 365 floats */
     char code[12];               /* array of 12 char */
     int health[];                /* wrong */
     int states[N];               /* array of 5 int */
     int ages[];                  /* wrong */
     int price[]  = {7, 3, 15};   /* array of 3 int */
     int debt[3] = {7, 3, 15};    /* array of 3 int */

}
```

Arrays – contd.

- Initializing arrays

```
int counters[5] = { 0, 0, 0, 0, 0 };
char letters[5] = { 'a', 'b', 'c', 'd', 'e' };
float sample_data[500] = { 100.0, 300.0, 500.5 };
```

- The C language allows you to define an array without specifying the number of elements. If this is done, the size of the array is determined automatically based on the number of initialization elements: `int counters[] = { 0, 0, 0, 0, 0 };`

Arrays – contd.

- Specifying the size of an array

```
# include <stdio.h>
# define m 5
# define n 3

void main()
{
      float a1[5];                    /* yes */
      float a2[2*5 + m + 1];          /* yes */
      float a3[n + m];                /* yes */
      float a4[2.5];                  /* no */
      float a5[0];                    /* no */
      float a6[ 8 / 2];               /* yes */
      float a7[ 8 / 3];               /* yes */
      float a8[ 8./ 3];               /* no */
      float a9[(int) 3.1];            /* yes */

}
```

Arrays – contd.

```
// Program to generate the first 15 Fibonacci numbers
#include <stdio.h>
int main (void)
{
  int Fibonacci[15], i;
  Fibonacci[0] = 0; // by definition
  Fibonacci[1] = 1; // ditto
  for ( i = 2; i < 15; ++i )
          Fibonacci[i] = Fibonacci[i-2] + Fibonacci[i-1];
  for ( i = 0; i < 15; ++i )
          printf ("%i\n", Fibonacci[i]);
  return 0;
}
```

Output

```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
```

Arrays – contd.

- Why don't we need the '&' before the array name?
  - Because the name of the array is an address

```
# include <stdio.h>

# define SIZE 6
void main()
{
   int ii = 0;
   int a[SIZE];

   for(ii = 0; ii < SIZE; ii++)
      printf("a[%d] %p\n", ii, &a[ii]);
}
    /* each integer is 4 bytes */
```

Output

```
a[0]      0012FF64
a[1]      0012FF68
a[2]      0012FF6C
a[3]      0012FF70
a[4]      0012FF74
a[5]      0012FF78
```

```
# include <stdio.h>

# define SIZE 6
void main()
{
   int ii = 0;
   int a[SIZE];

   for(ii = 0; ii < SIZE; ii++)
      printf("a[%d] %p\n", ii, a + ii);
}
    /* each integer is 4 bytes */
```

Output

```
a[0]      0012FF64
a[1]      0012FF68
a[2]      0012FF6C
a[3]      0012FF70
a[4]      0012FF74
a[5]      0012FF78
```

Arrays – contd.

- How can we generate random numbers?
  - We can use the functions rand() and srand() from the library stdlib.h

```
// generate random bits (0 or 1)
# include <stdio.h>
# include <stdlib.h>
# include <time.h>

# define N 10

void main()
{
      int array[N], ii = 0;
      // set the seed value (which number to start with)
      // the seed is set to the current time in seconds
      srand(time(NULL));

      for(ii = 0; ii < N; ii++)
          array[ii] = rand() % 2;

      for(ii = 0; ii < N; ii++)
          printf("array[%d] = %d\n", ii, array[ii]);
}
```

Output is random, and is different at every run

Output

```
array[0] = 0
array[1] = 1
array[2] = 0
array[3] = 0
array[4] = 1
array[5] = 0
array[6] = 1
array[7] = 1
array[8] = 0
array[9] = 1
```

Multidimensional arrays

- C language allows arrays of any number of dimensions
- Two-dimensional array: matrix

```
int M[4][5]; // matrix, 4 rows, 5 columns
//M[i][j] – element at row i, column j

int M[4][5] = {
          { 10, 5, -3, 17, 82 },
          { 9, 0, 0, 8, -7 },
          { 32, 20, 1, 0, 14 },
          { 0, 0, 8, 7, 6 }
};
int M[4][5] = { 10, 5, -3, 17, 82, 9, 0, 0, 8, -
7, 32,20, 1, 0, 14, 0, 0, 8, 7, 6 };
```

10	5	-3	17	82
9	0	0	8	-7
32	20	1	0	14
0	0	8	7	6

Example: Typical matrix processing

```
#define N 3
#define M 4
int main(void) {
    int a[N][M];
    int i,j;
    /* write matrix elements */
    for(i = 0; i < N; i++)
        for(j = 0; j< M; j++) {
            printf("a[%d][%d] = ", i, j);
            scanf("%d", &a[i][j]);
            }
    /* print matrix elements */
    for(i = 0; i < N; i++) {
        for(j = 0; j< M; j++)
            printf("%5d", a[i][j]);
        printf("\n");
        }
    return 0;
    }
```

Example: Dealing with variable numbers of elements

```c
#include <stdio.h>
#define NMAX 4

int main(void) {
    int a[NMAX];
    int n;
    int i;
    printf("How many elements(maximum %d)?\n",NMAX);
    scanf("%d",&n);
    if (n>NMAX) {
        printf("Number too big !\n");
        return 1; //this indicates the software terminate with error
        }
    for(i = 0; i < n; i++)
          scanf("%d", &a[i]);
    for(i = 0; i < n; i++)
          printf("%5d", a[i]);
    printf("\n");
    return 0;
    }
```

Variable length arrays

- A feature introduced by C99
- It was NOT possible in ANSI C !

```
int a[n];
```

- The array a is declared to contain n elements. This is called a variable length array because the size of the array is specified by a variable and not by a constant expression.
- The value of the variable must be known at runtime when the array is created => the array variable will be declared later in the block of the program

- Possible in C99: variables can be declared anywhere in a program, if the declaration occurs before the variable is first used.
- A similar effect of variable length array could be obtained in ANSI C using dynamic memory allocation to allocate space for arrays while a program is executing.

Example: Variable length arrays

```c
#include <stdio.h>

int main(void) {
    int n;
    int i;

    printf("How many elements do you have ? \n");
    scanf("%d",&n);

    int a[n];

    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n; i++)
        printf("%5d", a[i]);
    printf("\n");
    return 0;
}
```

Array a of size n created.
Value of n must be set at runtime before arriving at the array declaration !

Example: variable length arrays

```
#include <stdio.h>

int main(void) {
     int n;
     int i;

     n=7;
     int a[n];
     for(i = 0; i < n; i++)
           a[i]=i
     n=20;
     for(i = 0; i < n; i++)
           a[i]=2*i;
     printf("\n");
     return 0;
}
```

Array a created of size 7

Wrong!

Variable-length array does NOT mean that you can modify the length of the array after you create it ! Once created, a VLA keeps the same size !

Lecture Keywords

- Keywords
  - Memory address
  - Integer/double/float Array
  - continuo , break statements
  - goto statement
  - rand() and srand() functions
  - %p
  - Multidimension arrays
  - Variable size arrays

Lecture Summary

- break & continue Statements
- Introduced goto statements
- Introduced the single dimension arrays in C language
- Introduced the rand() and srand() functions
- Introduced the multidimension arrays in C language
- Introduced the variable size arrays in C language