Introduction to Scientific and Engineering Computation (BIL 104E)

Lab 9

Arrays: Initializing Arrays

Initializing an Array

```
1:  /* 12L01.c: Initializing an array */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int i;
7:     int list_int[10];
8:
9:     for (i=0; i<10; i++){
10:       list_int[i] = i + 1;
11:       printf( "list_int[%d] is initialized with %d.\n", i, list_int[i]);
12:    }
13:    return 0;
14: }
```

```
list_int[0] is initialized with 1.
list_int[1] is initialized with 2.
list_int[2] is initialized with 3.
list_int[3] is initialized with 4.
list_int[4] is initialized with 5.
list_int[5] is initialized with 6.
list_int[6] is initialized with 7.
list_int[7] is initialized with 8.
list_int[8] is initialized with 9.
list_int[9] is initialized with 10.
```

Arrays: The Size of an Array

Calculating the Size of an Array

```
1:  /* 12L02.c: Total bytes of an array */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int total_byte;
7:     int list_int[10];
8:
9:     total_byte = sizeof (int) * 10;
10:    printf( "The size of int is %d-byte long.\n", sizeof (int));
11:    printf( "The array of 10 ints has total %d bytes.\n", total_byte);
12:    printf( "The address of the first element: %p\n", &list_int[0]);
13:    printf( "The address of the last element:  %p\n", &list_int[9]);
14:    return 0;
15: }
```

The size of int is 4-byte long.
The array of 10 ints has total 40 bytes.
The address of the first element: 0022FF00
The address of the last element: 0022FF24

Arrays: Arrays and Pointers

Referencing an Array with a Pointer

```
1:  /* 12L03.c: Referencing an array with a pointer */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int *ptr_int;
7:     int list_int[10];
8:     int i;
9:
10:    for (i=0; i<10; i++)
11:       list_int[i] = i + 1;
12:    ptr_int = list_int;
13:    printf( "The start address of the array: %p\n", ptr_int);
14:    printf( "The value of the first element: %d\n", *ptr_int);
15:    ptr_int = &list_int[0];
16:    printf( "The address of the first element: %p\n", ptr_int);
17:    printf( "The value of the first element: %d\n", *ptr_int);
18:    return 0;
19: }
```

```
The start address of the array: 0022FF00
The value of the first element: 1
The address of the first element: 0022FF00
The value of the first element: 1
```

Arrays: Arrays and Pointers

```
#include <stdio.h>
main()
{
    int *ptr_int;
    int list_int[10];
    int i;
    for (i=0; i<10; i++)
        list_int[i] = i + 1;
    for (i=0; i<10; i++) {
        ptr_int = list_int;
        ptr_int += i;
        printf("list_int[%d] = %d \n",i , *ptr_int);
    }
    printf("\n");
    ptr_int = list_int;
    for (i=0; i<10; i++) {
        printf("list_int[%d] = %d \n",i , *ptr_int++);
    }
    getchar();
    return 0;
}
```

Arrays: Arrays and Pointers

```
list_int[0] = 1
list_int[1] = 2
list_int[2] = 3
list_int[3] = 4
list_int[4] = 5
list_int[5] = 6
list_int[6] = 7
list_int[7] = 8
list_int[8] = 9
list_int[9] = 10

list_int[0] = 1
list_int[1] = 2
list_int[2] = 3
list_int[3] = 4
list_int[4] = 5
list_int[5] = 6
list_int[6] = 7
list_int[7] = 8
list_int[8] = 9
list_int[9] = 10
```

Arrays: Displaying Arrays of Characters

Printing an Array of Characters

```
1:  /* 12L04.c: Printing out an array of characters */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     char array_ch[7] = {'H', 'e', 'l', 'l', 'o', '!', '\0'};
7:     int i;
8:
9:     for (i=0; i<7; i++)
10:       printf("array_ch[%d] contains: %c\n", i, array_ch[i]);
11:    /*---  method I ---*/
12:    printf( "Put all elements together(Method I):\n");
13:    for (i=0; array_ch[i] != '\0' && i<7; i++)
14:       printf("%c", array_ch[i]);
15:    /*---  method II ---*/
16:    printf( "\nPut all elements together(Method II):\n");
17:    printf( "%s\n", array_ch);
18:
19:    return 0;
20: }
```

Arrays: Displaying Arrays of Characters

Computer Screen

```
array_ch[0] contains: H
array_ch[1] contains: e
array_ch[2] contains: l
array_ch[3] contains: l
array_ch[4] contains: o
array_ch[5] contains: !
array_ch[6] contains:
Put all elements together(Method I):
Hello!
Put all elements together(Method II):
Hello!
```

Arrays: The Null Character ('\0')

Ending Output at the Null Character

```
1:  /* 12L05.c: Stopping at the null character */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     char array_ch[15] = {'C', ' ',
7:                          'i', 's', ' ',
8:                          'p', 'o', 'w', 'e', 'r',
9:                          'f', 'u', 'l', '!', '\0'};
10:    int i;
11:    /*  array_ch[i] in logical test */
12:    for (i=0; array_ch[i]; i++)
13:       printf("%c", array_ch[i]);
14:
15:    printf("\n");
16:    return 0;
17: }
```

```
C is powerful!
```

Arrays: Multidimensional Arrays

Printing a Two-Dimensional Array

```
1:  /* 12L06.c: Printing out a 2-D array */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     int two_dim[3][5] = {1, 2, 3, 4, 5,
7:                          10, 20, 30, 40, 50,
8:                          100, 200, 300, 400, 500};
9:     int i, j;
10:
11:    for (i=0; i<3; i++){
12:       printf("\n");
13:       for (j=0; j<5; j++)
14:          printf("%6d", two_dim[i][j]);
15:    }
16:    printf("\n");
17:    return 0;
18: }
```

```
     1     2     3     4     5
    10    20    30    40    50
   100   200   300   400   500
```

Arrays: Unsized Arrays

Initializing Unsized Arrays

```
1:  /* 12L07.c: Initializing unsized arrays */
2:  #include <stdio.h>
3:
4:  main()
5:  {
6:     char array_ch[] = {'C', ' ',
7:                        'i', 's', ' ',
8:                        'p', 'o', 'w', 'e', 'r',
9:                        'f', 'u', 'l', '!', '\0'};
10:    int list_int[][3] = {
11:           1, 1, 1,
12:           2, 2, 8,
13:           3, 9, 27,
14:           4, 16, 64,
15:           5, 25, 125,
16:           6, 36, 216,
17:           7, 49, 343};
18:
19:    printf("The size of array_ch[] is %d bytes.\n", sizeof (array_ch));
20:    printf("The size of list_int[][3] is %d bytes.\n", sizeof (list_int));
21:    return 0;
22: }
```

Arrays: Unsized Arrays

Computer Screen

```
The size of array_ch[] is 15 bytes.
The size of list_int[][3] is 84 bytes.
```