C Programming Basics

Ritu Arora

Texas Advanced Computing Center

June 19th, 2012

Email: rauta@tacc.utexas.edu

HW-3, Q-1 (page 1)

```
#include <stdio.h>
int main(){
 char myChar;
 int firstNum, secondNum;
 printf("My Calculator!");
 printf("\nEnter operation: +, -, *, /\n");
 scanf("%c",&myChar);
 printf("\nEnter first integer\n");
 scanf("%d", &firstNum);
 printf("\nEnter second integer\n");
 scanf("%d", &secondNum);
 printf("\ncharacter entered: %c\n", myChar);
 switch(myChar){
     case '+':
        printf("\nSum is: %d",(firstNum + secondNum));
        break;
```

HW-3, Q-1 (page 2)

```
    case '-':
        printf("\nDifference is: %d",(firstNum-secondNum));
        break;
     case '*':
        printf("\nProduct is: %d", (firstNum * secondNum));
        break;
     case '/':
        printf("\nQuotient is: %d", (firstNum/secondNum));
        break;
     default:
        printf("\nThis is not a valid operator.\n");
  }
  return 0;
}
```

HW-3, Q-2

```
#include <stdio.h>
int main(){
   int i, lowLimit, upLimit;
   int sum = 0;
   printf("\nEnter lower limit for the for-loop \n");
   scanf("%d", &lowLimit);
   printf("\nEnter upper limit for the for-loop \n");
   scanf("%d", &upLimit);
   for(i=lowLimit; i<= upLimit; i++){
      sum = sum + i;
   }
   printf("\nSum is: %d\n", sum);
   return 0;
}
```

Overview of the Course

- Writing a Basic C Program
- Understanding Errors
- Comments Keywords, Identifiers, Variables
- Operators
- Standard Input and Output (Basic)
- Control Structures
- Standard Input and Output
- Arrays, Structures
- Functions in C
- Pointers
- Working with Files

All the concepts will be accompanied with examples.

Arrays

- An array is a multivariable that allows you to store many different values of same data type in a single unit and in contiguous memory locations

- You can have arrays of any valid data type in C (not void though)

- Arrays are declared just like other variables, though the variable name ends with a set of square brackets
  - `char myName[50];` <---- You have seen this before
  - `int myVector[3];` //one-dimensional array
  - `int myMatrix[3][3];` //two-dimensional array

Initializing Arrays

- The content of the array is undetermined till you store any value in it

- Method 1

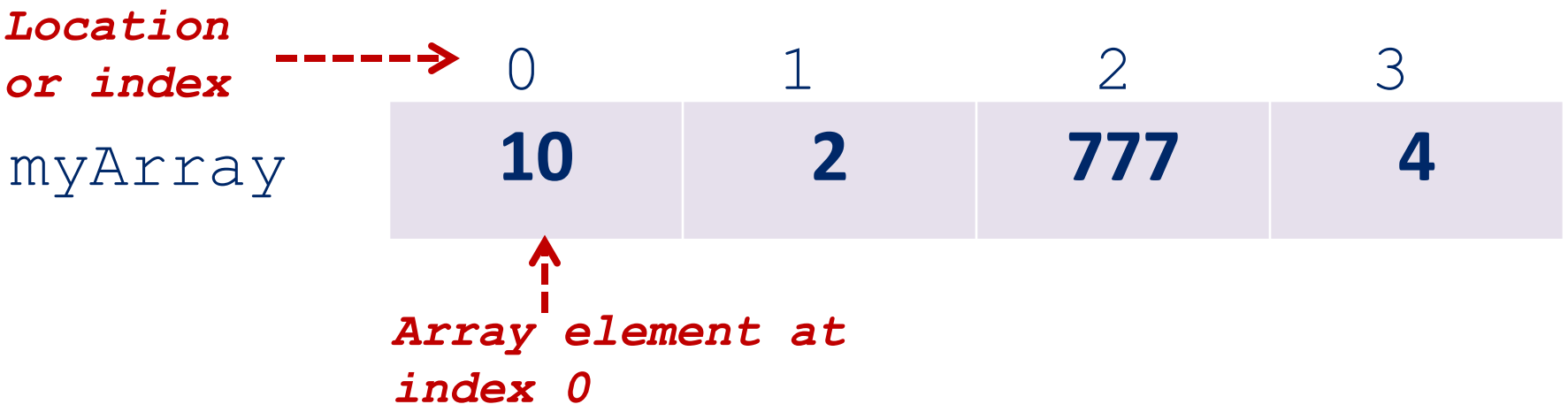
```
int myArray[4] = { 10, 2, 777, 4};
```

- Method 2

```
for (i =0; i<4; i++){
    scanf("%d",&myArray[i]);
}
```

Initializing Arrays

- Upon declaration and initialization an array is created like:




```
myArray[3]
myArray[1]
myArray[2]???
```

Computing With Arrays

- Access the array element

  `yArray[i]` where `i` is the index of the array

- Use it in computation like a regular variable

```
for (i=0; i< 3; i++){
   xArray[i] = yArray[i] + zArray[i];
}
xArray[i] = yArray[i] + zArray[i];
```

Arrays Example: arrayExample.c

```
#include <stdio.h>
int main(){
  int i;
  int age[4];
  age[0]=23;
  age[1]=34;
  age[2]=65;
  age[3]=74;
  for(i=0; i<4; i++){
    printf("age[%d]: %d\n", i, age[i]);
  }
  return 0;
}
```

Note: The number in the square brackets is the position number of a particular array element. Notice that count begins at 0

Output:

```
age[0]: 23
age[1]: 34
age[2]: 65
age[3]: 74
```

Multi-dimensional Arrays

- An array-of-arrays is called a multi-dimensional array
- A 2-dimensional array, `myArray` with 3 rows and 3 columns looks like:

	0	1	2
0	10 (0,0)	42 (0,1)	3 (0,2)
1	24	53	62
2	77	84	97

```
myArray[0][0] = 10;
myArray[1][2] = ???
```

2-D Arrays

```
1.  #include <stdio.h>
2.  int main(){
3.    int i, j;
4.    int xArray[2][2] = {{1, 2}, {3, 4}};
5.    int yArray[2][2] = {{1,2},{3,4}};
6.    int zArray[2][2] = {{0,0},{0,0}};
7.    for (i=0; i< 2; i++){
8.      for (j=0; j <2; j++){
9.         zArray[i][j] = xArray[i][j] + yArray[i][j];
10.     }
11.   }
12.   for (i=0; i< 2; i++){
13.     for (j=0; j <2; j++){
14.        printf(" %d ", zArray[i][j]);
15.     }
16.     printf("\n");
17.   }
18.   return 0;
19. }
```

Structures

- Multiple variables can be combined into a single package called structure
- Members of the structure variable need not be of the same type
- They can be used to do database work in C! Example:

```
struct sample{
   int a;
   char b;
};
struct sample mySample;
```

- `typedef` is the keyword that can be used to simplify the usage of `struct`

```
typedef struct sample newType;
```

Structure Example: structExample.c

```
#include <stdio.h>

typedef struct point{
  double x;
  double y;
}point;

int main(){
  point myPoint;
  myPoint.x = 12.2;
  myPoint.y = 13.3;
  printf("X is %lf and Y is %lf\n",myPoint.x, myPoint.y);
  return 0;
}
```

Overview of the Course

- Writing a Basic C Program
- Understanding Errors
- Comments Keywords, Identifiers, Variables
- Operators
- Standard Input and Output (Basic)
- Control Structures
- Standard Input and Output
- Arrays, Structures
- Functions in C
- Pointers
- Working with Files

All the concepts will be accompanied with examples.

C Language Functions

- Functions are self-contained blocks of statements that perform a specific task

- Written once and can be used multiple times
  - Promote code reuse
  - Make code maintenance easy

- Two steps involved
  - Write the function
    - Function definition
    - Function declaration or prototype
  - Invoke or call the function

- Two types of functions
  - Standard or library or built-in
  - User-Defined

Standard Functions

- These functions are provided to the user in library files
- In order to use the functions, the user should include the appropriate library files containing the function definition
- Example
  - `scanf`
  - `printf`
  - `gets`
  - `puts`
  - `strcpy`

User-Defined Functions: myFunction.c

```
#include <stdio.h>
---------------Defining the function add
void add(){
   int a, b, c;
   printf("\n Enter Any 2 Numbers : ");
   fflush(stdout);
   scanf("%d %d",&a,&b);
   c = a + b;
   printf("\n Addition is : %d",c);
}
int main(){
   add();  <-- Invoking the function add twice from
   add();  <-- function main
   return 0;
}
```

Function Prototype: myFctPrototype.c

```
#include <stdio.h>

void add();

int main(){
 add();
 return 0;
}

void add(){
int a, b, c;
printf("\n Enter Any 2 Numbers : ");
fflush(stdout);
scanf("%d %d",&a,&b);
c = a + b;
printf("\n Addition is : %d",c);
}
```

Function Prototype or Declaration: useful when the function is invoked before its definition is provided

Invoking the function add

Defining the function add that does not return a value – note void

Functions: Input and Output

- Functions that take no input, and return no output
- Functions that take input and use it but return no output
- Functions that take input and return output
- Functions that take no input but return output

Sending Input Values To Functions

- Determine the number of values to be sent to the function
- Determine the data type of the values that needs to be sent
- Declare variables having the determined data types as an argument to the function
- Use the values in the function
- Prototype the function if its definition is not going to be available before the place from where it is invoked
- Send the correct values when the function is invoked

Passing Values to Functions: passValue1.c

```
#include <stdio.h>
void add(int a, int b){<-- Formal Parameters: a, b
   int c;
   c = a + b;
   printf("\n Addition is : %d",c);
}
int main(){
   int a, b;
   printf("\n Enter Any 2 Numbers : ");
   fflush(stdout);
   scanf("%d %d",&a,&b);
   add(a, b); <-- Actual Parameters: a, b
   return 0;
}
```

Note: The variables used as formal and actual parameters can have different names.

Homework 5

- Write a program for matrix multiplication.
  - Declare a 2 X 2 array named `myMatrixA`
  - Declare a 2 X 2 array named `myMatrixB`
  - Declare a 2 X 2 array named `myMatrixC`
  - You can either initialize the array/matrix by reading the values from the keyboard or you can hard-code the values in the program
  - Note the formula:

```
myMatrixC[i][j] = myMatrixC[i][j] +
myMatrixA[i][k] * myMatrixB[k][j];
```

  - Write nested for-loops to find the product of `myMatrixA` and `myMatrixB` and store it in `myMatrixC`

References

- C Programming Language, Brian Kernighan and Dennis Ritchie
- Let Us C, Yashavant Kanetkar
- C for Dummies, Dan Gookin
- http://cplusplus.com