

C Programming II

Alexander B. Pacheco LTS Research Computing June 3, 2015

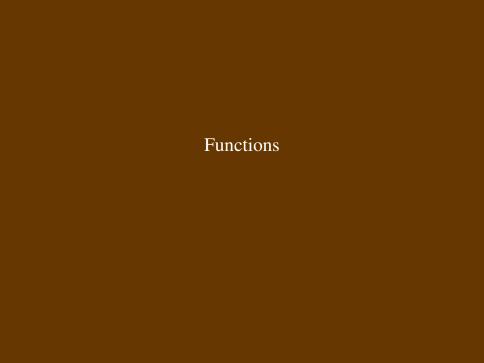
Outline

Functions

2 Arrays

3 Pointers

Input/Output



Functions

- A function is a group of statements that together perform a task.
- Every C program has at least one function, which is main()
- Functions receive either a fixed or variable amount of arguments.
- Functions can only return one value, or return no value (void).
- In C, arguments are **passed by value** to functions
- How to return value? Pointers
- Functions are defined using the following syntax:

```
return_type function_name( parameter list )
{
  body of the function
}
```

- A function declaration tells the compiler about a function's name, return type, and parameters.
- A function **definition** provides the actual body of the function.

Function Definition

- **Return Type:** Function's return type is the data type of the value the function returns. When there is no return value, return void.
- Function Name: This is the actual name of the function.
- **Parameter:** The parameter list refers to the type, order, and number of the parameters of a function. A function may contain no parameters.
- Function Body: The function body contains a collection of statements that define the function behavior.

Example of using a Function

```
#include <stdio.h>
/* function declaration */
int max(int i, int j);
int main() {
  /* local variable definition */
  int i = 100, j = 200, maxval;
 /* calling a function to get max value */
 maxval = max(a, b):
  printf( "Max value is : %d\n", maxval );
  return 0:
/* function returning the max between two numbers */
int max(int i, int j)
 /* local variable declaration */
  int result:
 if (i > j)
    result = i;
  else
    result = j;
  return result;
```

Scope Rules: Local & Global Variables I

- A scope is a region of the program where a defined variable can have its existence and beyond that variable can not be accessed.
- Local Variables: declared inside a function or block.
 can be used only by statements that are inside that function or block of code.
 Local variables are not known to functions outside their own.
- Global Variables: defined outside of a function, usually on top of the program.
 will hold their value throughout the lifetime of your program and,
 they can be accessed inside any of the functions defined for the program.
- A program can have same name for local and global variables but value of local variable inside a function will take preference.

Scope Rules: Local & Global Variables II

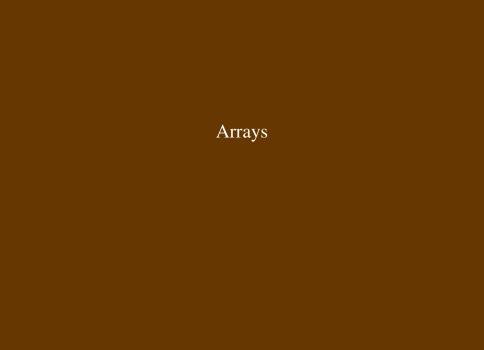
```
#include <stdio.h>
/* global variable declaration */
int a = 20;
int main ()
 /* local variable declaration in main function */
 int. a = 10:
 int b = 20;
 int c = 0;
 printf ("value of a in main() = %d\n", a);
 c = sum(a, b);
 printf ("value of c in main() = %d\n", c);
  return 0;
/* function to add two integers */
int sum(int a, int b)
 printf ("value of a in sum() = %d\n", a);
 printf ("value of b in sum() = %d\n", b);
  return a + b;
   value of a in main() = 10
   value of a in sum() = 10
    value of b in sum() = 20
    value of c in main() = 30
```

Initializing Local & Global Variables

- Local Variables are not initialized by the system, the programmer must initialize it.
- Global variables are automatically initialized by the system depending on the data type

Data Type	Initial Default Value
int	0
char	'\0'
float	0
double	0
pointer	NULL

• It is a good programming practice to initialize variables properly otherwise, your program may produce unexpected results because uninitialized variables will take some garbage value already available at its memory location.



Arrays

- Arrays are special variables which can hold more than one value using the same name with an index.
- Declaring Arrays: type arrayName [arraySize];

```
/* simply define the arrays */
double balance[10];
float atom[1000];
int index[5];
```

• C array starts its index from 0

[0]	[1]	[2]	[3]	[4]
10	15	14	3	7

index[2] (3rd element of the array) has a value 14

• Initialize arrays with values

```
/* initialize the array with values*/
double atmass[4] = {12.0, 1.0, 1.0, 16.0};
double atmass[] = {12.0, 1.0, 1.0, 16.0};
atmass[0] = 12.0
```

Access array values via index

```
/* access the array values*/
int current_index = index[i];
double current_value=value[current_cell_index];
```

Array Example

```
#include <stdio.h>
int main ()
{
    int n[ 10 ]; /* n is an array of 10 integers */
    int i, j;

/* initialize elements of array n to 0 */
    for ( i = 0; i < 10; i++)
    {
        n[ i ] = i + 100; /* set element at location i to i + 100 */
    }

/* output each array element's value */
    for (j = 0; j < 10; j++)
    {
        printf("Element[%d] = %d\n", j, n[j] );
    }

return 0;
}</pre>
```

Accessing C arrays

- C arrays are a sequence of elements with contiguous addresses.
- There is no bounds checking in C.
- Be careful when accessing your arrays
- Compiler will not give you error, you will have *undefined* runtime behavior:

```
#include <stdio.h>
int main() {
   int index[5]={5, 4, 6, 3, 1};
   int a=3;
   /* undefined behavior */
   printf("%d\n",index[5]);
```

Multidimensional Arrays

• General form of multidimensional array

```
type name[size1][size2]...[sizeN];
```

• Declaring 2D and 3D arrays:

```
float array2d[4][5];
double array3d[2][3][4];
```

Initializing multidimensional arrays

	col0	col1	col2	col3
row0	a[0][0]=0	a[0][1]=1	a[0][2]=2	a[0][3]=3
row1	a[1][0]=4	a[1][1]=5	a[1][2]=6	a[1][3]=7
row2	a[2][0]=8	a[2][1]=9	a[2][2]=10	a[2][3]=11

• C arrays are **row major** order i.e. in memory, the C array appears as

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[1][0]	a[1][1]	 a[1][3]	a[2][0]	 a[2][3]

Example: Arrays

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int main () {
 /* Program to calculate the sum, min and max of an integer array */
  int i. sum. min. max. n=11 :
  int a[] = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\};
  sum = max = 0.0 : min = 10.0 :
  /* Initialize array */
  for (i = 0 : i < n : i++) {
   sum += a[i] :
   if (a[i] > max ) max = a[i];
   if (a[i] < min ) min = a[i];</pre>
  printf("The max value is: %d\n", max);
  printf("The min value is: %d\n", min);
  printf("The sum value is: %d\n", sum);
  return 0:
/* define string */
char str[7]={'H', 'E', 'L', 'L, 'O', '!', '\0'};
```

Strings in C I

 Strings in C are a special type of array: array of characters terminated by a null character '\0'.

```
/* define string */
char str[7]={'H', 'E', 'L', 'L', 'O', '!', '\0'};
char strl="HELLO!";
```

• Memory presentation of above defined string in C/C++:

str[]	[0]	[1]	[2]	[3]	[4]	[5]	[6]
	'H'	'E'	'L'	'L'	'O'	'!'	'\0'

• C uses built-in functions to manipulate strings:

```
/* C sample string functions */
strcpy(s1, s2); /* Copies string s2 into string s1.*/
strcat(s1, s2); /* Concatenates string s2 onto the end of string s1. */
strlen(s1); /* Returns the length of string s1. */
strcmp(s1, s2); /* Returns 0 if s1 and s2 are the same; less than 0 if
s1<s2; greater than 0 if s1>s2. */
```

Strings in C II

```
#include <stdio.h>
#include <string.h>
int main ()
  char str1[12] = "Hello";
  char str2[12] = "World";
  char str3[12];
  int len;
  strcpy(str3, str1);
  printf("strcpy( str3, str1) : %s\n", str3 );
  strcat( strl, str2);
  printf("strcat( strl, str2): %s\n", strl );
  len = strlen(strl):
  printf("strlen(strl) : %d\n", len );
  return 0:
```

