

CSE1301
Computer Programming
Lecture 14
Functions (Part 2)

1

- Topics**
- Prototypes
 - Scope
- 2

- Prototyping of Functions**
- Must declare functions before use (like variables)
 - Declaration is called a “**prototype**”
 - Specifies the **name**, **parameters** and **return type** of the function, but not the code

Example: isNegative.c

<pre>#include <stdio.h> int isNegative (int); int main (void) { int number; printf ("Enter an integer: "); scanf ("%d",&number); if (isNegative(number)) { printf("Negative\n"); } else { printf("Positive\n"); } return 0; }</pre>	<pre>int isNegative (int n) { int result; if (n<0) { result=1; } else { result = 0; } return result; }</pre>
--	---

Example: isNegative.c

<pre>#include <stdio.h> int isNegative (int); int main (void) { int number; printf ("Enter an integer: "); scanf ("%d",&number); if (isNegative(number)) { printf("Negative\n"); } else { printf("Positive\n"); } return 0; }</pre>	<pre>int isNegative (int n) { int result; if (n<0) { result=1; } else { result = 0; } return result; }</pre>
--	---

Function Prototype

Example: isNegative.c

<pre>#include <stdio.h> int isNegative (int); int main (void) { int number; printf ("Enter an integer: "); scanf ("%d",&number); if (isNegative(number)) { printf("Negative\n"); } else { printf("Positive\n"); } return 0; }</pre>	<pre>int isNegative (int n) { int result; if (n<0) { result=1; } else { result = 0; } return result; }</pre>
--	---

Function Definition

Example: isNegative.c

```
#include <stdio.h>
int isNegative (int);
int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);

    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
}
return 0;
```

Function Call
(Must be after prototype, but can be before definition)

```
int isNegative ( int n )
{
    int result;
    if ( n<0 )
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}
```

Example: isNegative.c

```
#include <stdio.h>
int isNegative (int);
int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);

    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
}
return 0;
```

Header files (filename.h) contain function prototypes and global variable declarations

Example: isNegative.c

```
#include <stdio.h>
int isNegative (int);
int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);

    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
}
return 0;
```

stdio.h contains function prototypes for printf(), scanf(), and other I/O functions

```
if ( n<0 )
{
    result=1;
}
else
{
    result = 0;
}
return result;
```

Header files

- You can make your own header files with prototypes of frequently used functions:
#include "myFunctions.h"
- Put the functions in a corresponding C file, and include those too:
#include "myFunctions.c"

Example: isNegative.c

```
#include <stdio.h>
#include "myFunctions.h"
#include "myFunctions.c"
int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);

    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
}
return 0;
```

Note:
• " " around file name for user-defined files
• < > for standard system files

isNegative() is declared in myFunctions.h and defined in myFunctions.c, not in this file!

Example: myFunctions.c

```
int isNegative ( int n )
{
    int result;
    if ( n<0 )
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}
```

Example: myFunctions.h

```
int isNegative ( int );
```

Scope

Where can you use a variable which is declared in a function?

- In that function **only**
- Not in a *calling* function
- Not in a *called* function

Scope: Local Variables

- Formal parameters: only accessible whilst function executing
- Variables declared in a function body: only accessible whilst function executing
- In fact, this is true of every block in a program

Example: isNegative.c

```
#include <stdio.h>

int
isNegative ( int n )
{
    int result;
    if ( number<0)
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}

int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
    return 0;
}
```

Example: isNegative.c

```
#include <stdio.h>

int
isNegative ( int n )
{
    int result;
    if ( number<0)
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}

int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
    return 0;
}
```

ERROR! Number is local to the main function, not accessible here

Example: isNegative.c

```
#include <stdio.h>

int
isNegative ( int n )
{
    int result;
    if ( n<0)
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}

int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
    return 0;
}
```

Use the parameter **n** which is local to the function **isNegative()**

Example: isNegative.c

```
#include <stdio.h>

int
isNegative ( int n )
{
    int result;
    if ( n<0)
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}

int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
    return 0;
}
```

n is the *formal* parameter

number is the *actual* parameter

Example: isNegative.c

```
#include <stdio.h>

int
isNegative ( int n )
{
    int result;
    if ( n < 0 )
    {
        result=1;
    }
    else
    {
        result = 0;
    }
    return result;
}

int main (void)
{
    int number;
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    if (isNegative(number))
    {
        printf("Negative\n");
    }
    else
    {
        printf("Positive\n");
    }
}
```

*result & n: local to isNegative()
number: local to main()*

Example: isNegativeGlobal.c

```
#include <stdio.h>

int number;

int main (void)
{
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    isNegative ( void )
    {
        int result;
        if ( number < 0 )
        {
            result=1;
        }
        else
        {
            printf("Negative\n");
        }
    }
    else
    {
        printf("Positive\n");
    }
    result = 0;
}
return result;
}
```

Example: isNegativeGlobal.c

```
#include <stdio.h>

int number;

int main (void)
{
    printf ("Enter an integer: ");
    scanf ("%d",&number);
    isNegative ( void )
    {
        int result;
        if ( number < 0 )
        {
            result=1;
        }
        else
        {
            printf("Negative\n");
        }
    }
    else
    {
        printf("Positive\n");
    }
    result = 0;
}
return result;
}
```

*number is now GLOBAL -
declared outside any function,
accessible in all functions
(after the declaration)*

Scope: Global Variables

- Global variables are accessible in any function **after** their declaration to the end of that source file
- They're useful, but risky
 - if any and every function can modify them, it can be difficult to keep track of their value
- Better to use local variables and parameter passing if possible

Scope: Functions

- Functions are also accessible in any function **after** their declaration to the end of that source file

Summary

- Include function *prototype* **before** its use
- Be careful about the *scope* of variables

Readings

- King: 9.1 – 9.4
- D&D: 5.6 – 8.5, 5.12
- Kernighan & Ritchie: 4.1, 4.2, 4.4, 4.11