Lecture 10

Functions

CSE115: Computing Concepts

Functions with Multiple Parameter

- A function can have any number of parameters (zero or more)
- We have seen functions with no parameter (recall draw)

```
void draw_rectangle()
{
        printf(" ****** \n");
        printf(" *    * \n");
        printf(" *    * \n");
        printf(" *    * \n");
        printf(" ****** \n");
}
```

- We have seen function with one parameter

```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1;i<=x;i++)
        fact *= i;
    return fact;
}
```

Functions with Multiple Parameter

- Say we want to write a function that calculates a^b, where both a and b are positive integers

Functions with Multiple Parameter

```
#include <stdio.h>
int power(int a, int b);
int main()
{
        int x = 2, y = 8, z;
        z = power(x, y);
       printf("%d\n",z);
        return 0;
}
int power(int a, int b)
{
    int i, result = 1;
    for(i=1; i<=b; i++)
    {
        result *= a;
    }
    return result;
}
```

Scopes

```
#include <stdio.h>
void power(int a, int b);
int main()
{
        int x = 2, y = 8, z;
        power(x, y);
       printf("%d\n",result);
        return 0;
}
void power(int a, int b)
{
    int i, result = 1;
    for(i=1; i<=b; i++)
    {
        result *= a;
    }
}
```

Can we do this?

No, `result` is defined here (local to the `power` function), not inside main

Scopes

```
#include <stdio.h>
void power(int a, int b);
int main()
{
        int x = 2, y = 8, z;
        power(x, y);
       printf("%d\n",result);
        return 0;
}
void power(int a, int b)
{
    int i, result = 1;
    for(i=1; i<=b; i++)
    {
        result *= a;
    }
}
```

Different scopes. Variable from one scope is not visible inside another

Local variable

Scopes

```
#include <stdio.h>
void power(int a, int b);
int result;
int main ()
{
    int x = 2, y = 8, z;
    power(x, y);
    printf("%d\n",result);
    return 0;
}
void power(int a, int b)
{
    int i;
    for(i=1, result=1; i<=b; i++)
    {
        result *= a;
    }
}
```

How about this?

Scopes

```
#include <stdio.h>
void power(int a, int b);
int result;
int main ()
{
    int x = 2, y = 8, z;
    power(x, y);
    printf("%d\n",result);
    return 0;
}
void power(int a, int b)
{
    int i;
    for(i=1, result=1; i<=b; i++)
    {
        result *= a;
    }
}
```

How about this?

Scopes

```
#include <stdio.h>
void power(int a, int b);
int result;
int main ()
{
    int x = 2, y = 8, z;
    power(x, y);
    printf("%d\n",result);
    return 0;
}
void power(int a, int b)
{
    int i;
    for(i=1, result=1; i<=b; i++)
    {
        result *= a;
    }
}
```

Global variable (declared in the global scope) is visible in both scopes

An Example

- Lets find if N is a prime number, where N will be provided by the user

An Example

```
int is_prime(int a)
{
    int i;
    for(i=2; i<a; i++)
    {
        if(a%i == 0)
            return 0;
    }
    return 1;
}
```

Another Example

Take a number N as input from the user and print this pyramid of asterisk. N is the number of asterisk in the last row (you can assume that N will always be a positive odd number).

```
    *
   ***
  *****
 *******
*********
```

Another Example

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Another Example

Will it work?

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Another Example

Will it work?

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Output for n = 9:

```
*
***
*****
*******
*********
```

Another Example

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Will it work?

We should print spaces before doing this

Output for n = 9:

```
*
***
*****
*******
*********
```

Another Example

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Another Example

```
void pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Output for n = 9:

```
    *
   ***
  *****
 *******
*********
```

Yet Another Example

Take a number N as input from the user and print this diamond of asterisk. N is the number of asterisk in the middle row (you can assume that N will always be a positive odd number).

```
      *
     ***
    *****
   *******
*********
*********
   *******
    *****
     ***
      *
```

Yet Another Example

```
void upper_pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

```
void lower_pyramid(int n)
{
    int i, j;
    for(i=n; i>=1; i-=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Yet Another Example

```
void upper_pyramid(int n)
{
    int i, j;
    for(i=1; i<=n; i+=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

```
void lower_pyramid(int n)
{
    int i, j;
    for(i=n; i>=1; i-=2)
    {
        for(j=1; j<=(n-i)/2; j++)
        {
            printf(" ");
        }
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 1;
}
```

Home-works

Implement the following functions.

1. `int sum(int n);`

Description: Returns the sum $1 + 2 + \cdots + n$

2. `void listNumbersAsc(int start, int end);`

Description: Outputs the numbers from start to end in ascending order.

3. `void listNumbersDesc(int start, int end);`

Description: Outputs the numbers from start to end in descending order.

4. `double harmonicSum(int n);`

Description: Returns the sum $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

5. `int sumOfDigits(int x);`

Description: Returns the sum of digits of the positive integer x. For example, when called with the parameter 12345, this function returns 15.