Functions

Nalini Vasudevan
Columbia University

Usage

- To avoid repetitive code
- Written once, can be instantiated multiple times
- We have already seen
  - `main, printf`

Let's multiply two numbers

Multiplying two numbers

In Mathematics:

f(x, y) = x * y;

Multiplying two numbers

```
f(x, y)
{
     x * y;
}
```

Multiplying two numbers

```
int f(int x, int y)
{
    x * y;
}
```

Multiplying two numbers

```
int f(int x, int y)
{
    z = x * y;
    return z;
}
```

Multiplying two numbers

```
int f(int x, int y)
{
        int z;
     z = x * y;
     return z;
}
```

Multiplying two numbers

```
int multiply(int x, int y)
{
        int z;
    z = x * y;
    return z;
}

main ()
{
    int a, b, c;
    a = b = 26;
    c = multiply (a, b);
    printf("Answer = %d", c);
}
```

Multiplying two numbers

```
int multiply(int, int); /*prototype
                          declaration*/
main ()
{
        int a, b, c;
        a = b = 26;
        c = multiply (a, b);
        printf(“Answer = %d”, c);
}

int multiply(int x, int y)
{
        z = x * y;
        return z;
}
```

Multiplying two numbers

```
main ()
{
        int a, b, c, p, q, r;
        a = b = q = 15;
        p = 12;
        c = multiply (a, b);
        r = multiply (p, q);
        printf(“Answer = %d, %d”, c, r);
}
```

Scope

```
void change (float x)
{
     x = x/2;
     printf(“%f”, x);
}

main()
{
     float x = 9;
     change(x);
     printf(“%f”, x);
}
```

Scope

```
void change (float x)
{
     x = x/2;
     printf(“%f”, x); /* 4.5 */
}

main()
{
     float x = 9;
     change(x);
     printf(“%f”, x); /* 9 */
}
```

Scope

```
float x = 9; /* global variable */

void change ()
{
    x = x/2;
    printf(“%f”, x);
}

main()
{
    change();
    printf(“%f”, x);
}
```

Scope

```
float x = 9; /* global variable */

void change ()
{
    x = x/2;
    printf("%f", x); /* 4.5 */
}

main()
{
    change();
    printf("%f", x); /* 4.5 */
}
```

Scope

```
float x = 9; /* global variable */

void change (float x)
{
     x = x/2;
     printf(“%f”, x);
}

main()
{
     change(x);
     printf(“%f”, x);
}
```

Scope

```
float x = 9; /* global variable */

void change (float x)
{
     x = x/2;
     printf("%f", x); /* 4.5*/
}

main()
{
     change(x);
     printf("%f", x); /* 9 */
}
```

Scope

```
float x = 9; /* global variable */

void change ()
{
        x = x/2;
        printf(“%f”, x);
}

main()
{
        float x = 5;
        change();
        printf(“%f”, x);
}
```

Scope

```
float x = 9; /* global variable */

void change ()
{
        x = x/2;
        printf(“%f”, x); /* 4.5 */
}

main()
{
        float x = 5;
        change();
        printf(“%f”, x); /* 5 */
}
```

Scope

```
float x = 9; /* global variable */

void change (float x)
{
        x = x/2;
        printf(“%f”, x);
}

main()
{
        float x = 5;
        change(x);
        printf(“%f”, x);
}
```

Scope

```
float x = 9; /* global variable */

void change (float x)
{
     x = x/2;
     printf(“%f”, x); /* 2.5 */
}

main()
{
     float x = 5;
     change(x);
     printf(“%f”, x); /* 5 */
}
```

Scope

```
main ()
{
   int x = 5;
   if (x)
   {      int x = 10;
          x++;
          printf (“%d”, x);
   }
   x++;
   printf (“%d”, x);
}
```

Scope

```
main ()
{
   int x = 5;
   if (x)
   {      int x = 10;
          x++;
          printf ("%d", x); /* 11 */
   }
   x++;
   printf ("%d", x); /* 6 */
}
```

Arrays and Strings

```
int strlen (char s[])
{
   int i, len = 0;
   for (i = 0; s[i] != '\0'; i++)
         len++;
   return len;
}
main()
{
   char str[10];
   strcpy(str, “Nalini”);
   printf(“%d”, strlen(str));
}
```

Recursion

```
void change (count)
{
        ..
        ..
        change(count);
        ..
}
```

A function calls itself

Sum of elements of an array

```
int sum (int a[], int size)
{
     int i, result = 0;
     for (i = 0; i < size; i++)
     {
          result +=  a[i];

     }
     return result;
}
```

Sum using Recursion

```
int a[] = {10, 9, 8, 7, 6, 5, 4};


sum (a, 7) =  10 + 9 + 8 + 7 + 6 + 5 + 4;
           =  (10 + 9 + 8 + 7 + 6 + 5) + 4;
       =  sum (a, 6) + 4;


sum (a, 6) =  10 + 9 + 8 + 7 + 6 + 5;
           =  (10 + 9 + 8 + 7 + 6) + 5;
       =  sum (a, 5) + 5;
```

Sum using Recursion

```
int a[] = {10, 9, 8, 7, 6, 5, 4};


sum (a, 2) =  10 + 9;
         =  (10) + 9;
       =  sum (a, 1) + 9;


sum (a, 1) =  10;
           = (0) + 10;
       =  sum (a, 0) + 10;

sum (a, 0) = 0;
```

Sum using Recursion

```
Generalizing:

sum (a, n) = sum (a, n-1) + a[n]; if n > 0
sum (a, 0) = 0 ; otherwise
```

Sum using Recursion

```
sum (a, n) = sum (a, n-1) + a[n]; if n > 0
sum (a, 0) = 0 ; otherwise
```

```
int sum (int a[], int n)
{
   if (n > 0)
   {
      int result;
      result = sum (a, n- 1);
      result += a[n-1];
      return result;
   }
   else
         return 0;
}
```

Sum using Recursion

```
sum (a, n) = sum (a, n-1) + a[n]; if n > 0
sum (a, 0) = 0 ; otherwise
```

```
int sum (int a[], int n)
{
   if (n == 0)
        return 0;
   return sum (a, n-1) + a[n-1];
}
```

Fibonacci series

1, 1, 2, 3, 5, 8, 13, 21......

Problem: Print the nth Fibonacci number

```
fib (1) = 1;
fib (2) = 1;
fib (4) = 3;
fib (7) = 13;
```

Fibonacci series

```
int fib(int n)
{
    if (n == 1 || n == 2)
    {
        return 1;
    }
    return fib(n-1) + fib(n-2);
}
```

Static Variables

```
void increment()
{
    static int i = 0;
    i++;
    printf(“%d”, i);
}

main()
{
        increment();
        increment();
        increment();
}
```

Puzzle

```
int  i = 4;

void compute()
{
     static int i = 0;
     for (; i < 3; i ++)
          printf("%d\n", i);
}

main()
{
  while (i --)
       compute();
  printf("%d\n", i);
}
```

Register Variables

- Placed in registers
  - faster code

```
void compute(int n)
{
        register int i;
        for (i = 0; i < n; i++)
        {
                /* Do something */
        }
}
```

C Preprocessor

```
#define MAX(A, B) (A > B ? A : B)

int largest(int a, int b, int c)
{
    int result;
    result = MAX(a, b);
    result = MAX(result, c);
    return result;
}
```

Puzzle

```
#define A  30
#define B  20
#define C  A - B

main()
{
   int r;
   r = C * 30;
   printf("%d\n", r);
}
```