



Today:

Functions



Some General Tips on Programming

- Write your code modularly
- Compile + test functionality in the process



Some General Tips on Programming (Cont.)

- Use Debugger or printf to follow your execution flow and find what went wrong
- Understanding is good but not enough – you must practice!



Example: Factorial

```
#include <stdio.h>
```

```
int factorial(int n)
```

```
{
```

```
    int i, fact = 1;
```

```
    for (i=2; i<=n; i++)
```

```
        fact *= i;
```

```
    return fact;
```

```
}
```

```
int main(void)
```

```
{
```

```
    int num;
```

```
    printf("enter a number\n");
```

```
    scanf("%d",&num);
```

```
    printf("%d!=%d\n",num,factorial(num));
```

```
    return 0;
```

```
}
```



Exercise: Fibonacci (@class)

- Input:
 - An integer n
- Output:
 - The n 'th fibonacci number
- Note –
 - Use an appropriately defined function



Solution

fibonacci_func.c



A Detailed Example

Write a program that receives a nominator and a denominator from the user, and displays the reduced form of the number.

For example, if the input is 6 and 9, the program should display $2/3$.



Example – solution (step I)

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int n, d;
```

```
    printf("Please enter nominator and denominator: ");
```

```
    scanf("%d%d", &n, &d);
```

Calculate n's and d's Greatest Common Divisor

```
    printf("The reduced form of %d/%d is %d/%d", n, d, n/gcd, d/gcd);
```

```
    return 0;
```

```
}
```




Example – solution (step II)

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int n, d, g;
```

```
    printf("Please enter nominator and denominator: ");
```

```
    scanf("%d%d", &n, &d);
```

```
    g = gcd(n, d);
```

```
    printf("The reduced form of %d/%d is %d/%d", n, d, n/g, d/g);
```

```
    return 0;
```

```
}
```



Example – solution (step III)

```
/* Returns the greatest common divisor of its  
two parameters. Assumes both are positive.  
*/
```

```
int gcd(int x, int y)
```

```
{
```

```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```



GCD – step by step

```
int main(void)
```

```
{
```

```
    int n, d, g;
```

```
    printf("Please enter ... : ");
```

```
    scanf("%d%d", &n, &d);
```

```
    g = gcd(n, d);
```

```
    printf("The reduced form...", n, d, n/g, d/g);
```

```
    return 0;
```

```
}
```

n	d	g
6	9	---



GCD – step by step

```
int main(void)
```

```
{
```

```
    int n, d, g;
```

```
    printf("Please enter ... : ");  
    scanf("%d%d", &n, &d);
```

```
    g = gcd(n, d);
```

```
    printf("The reduced form...", n, d, n/g, d/g);
```

```
    return 0;
```

```
}
```

n	d	g
6	9	---



GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

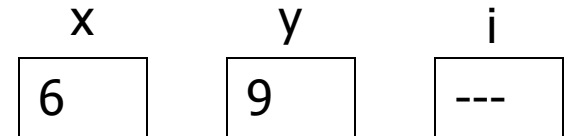
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

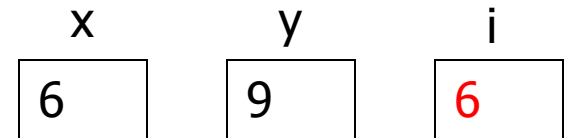
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

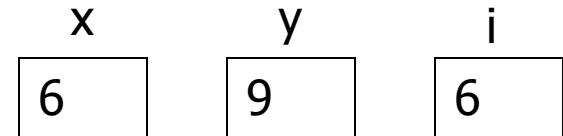
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

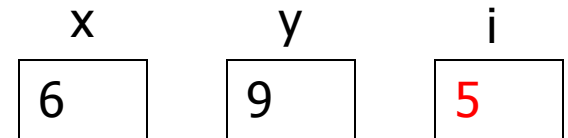
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

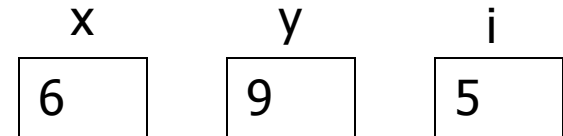
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

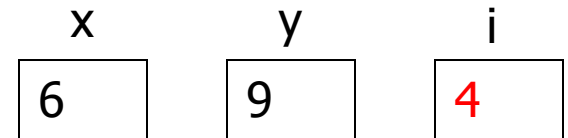
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

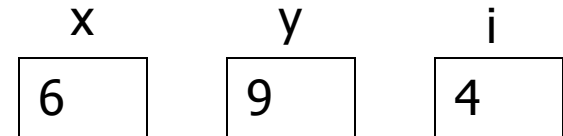
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

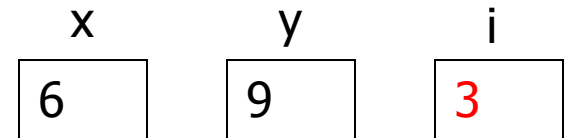
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

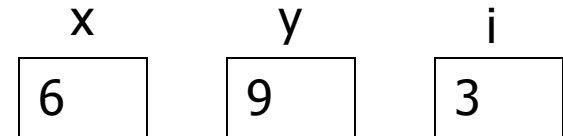
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int gcd(int x, int y)
```

```
{
```

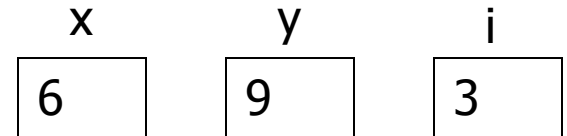
```
    int i;
```

```
    for (i=x; i>0; i--)
```

```
        if (x%i == 0 && y%i == 0)
```

```
            return i;
```

```
}
```





GCD – step by step

```
int main(void)
```

```
{
```

```
    int n, d, g;
```

```
    printf("Please enter ... : ");  
    scanf("%d%d", &n, &d);
```

```
    g = gcd(n, d);
```

```
    printf("The reduced form...", n, d, n/g, d/g);
```

```
    return 0;
```

```
}
```

n	d	g
6	9	3



Example

- Newton was the first to notice that for any positive n , and when $x_0=1$, the following series converges to $\text{sqrt}(n)$ –

$$x_{k+1} = \frac{1}{2} \left(x_k + \frac{n}{x_k} \right)$$

- Use this fact to write a program that accepts a positive number and outputs its square root
 - Hint – the thousandth element of Newton's series is a good-enough approximation



sqrt - solution

```
/* This program uses the Newton method to calculate a number's  
square root */
```

```
#include <stdio.h>
```

```
double sqrt(double num) {
```

```
    int i;
```

```
    double result = 1;
```

```
    for (i=0; i<1000; i++)
```

```
        result = (result + num/result)/2;
```

```
    return result;
```

```
}
```



sqrt – solution cont.

```
int main () {
    double num;

    printf("Please enter positive number: ");
    scanf("%lf", &num);

    if (num < 0) {
        printf("There's no square root for a negative number!\n");
        return 1;
    }
    printf("The square root of %g is %g\n", num, sqrt(num));
    return 0;
}
```



Pass-By-Value: Example

```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

add_one – step by step

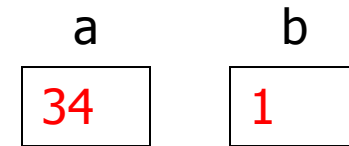
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one – step by step

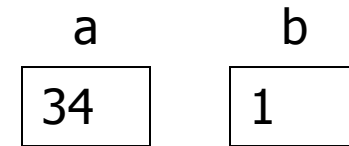
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one – step by step

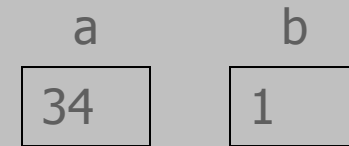
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

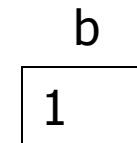
    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one memory state



add_one – step by step

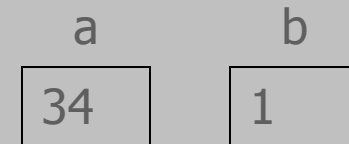
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

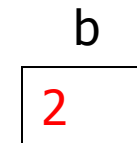
    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one memory state



add_one – step by step

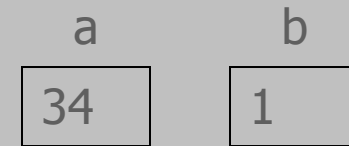
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

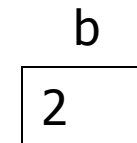
    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one memory state



add_one – step by step

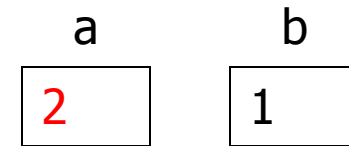
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state



add_one – step by step

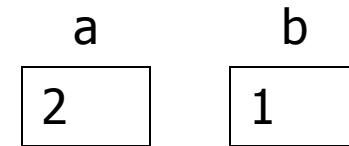
```
int add_one(int b)
{
    b=b+1;
    return b;
}
```

```
int main(void)
{
    int a=34,b=1;

    a=add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

Main() memory state





Example: Error in Scope

```
int add_one(int b)
{
    a=b+1;
    return a;
}
```

```
int main(void)
{
    int a=34,b=1;

    add_one(b);

    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```



Example

Write a function that uses the formula

$$\frac{\pi^2}{6} = 1/1 + 1/4 + 1/9 + 1/16 + \dots + 1/n^2$$

(where n goes to infinity)

in order to approximate π . The function should accept an argument n which determines the number of terms in the formula. It should return the approximation of π .

Write a program that gets an integer n from the user, and approximate π using n terms of the above formula.



pi - solution

```
#include <math.h>
```

```
/* Returns an approximation of pi using num terms */
```

```
double pi(int num) {  
    int n;  
    double result = 0.0;  
    for (n = 1; n <= num; n++)  
        result += 1.0 / (n * n);  
  
    return sqrt(6*result);  
}
```



pi – solution cont.

```
/*This program approximates pi*/
```

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int num;
```

```
    printf("enter number of terms\n");
```

```
    scanf("%d",&num);
```

```
    printf("The calculated approximation of pi is: %g\n",pi(num));
```

```
    return 0;
```

```
}
```



Example

Modify the previous function that approximates π . The function should accept an argument specifying the desired accuracy, and keep adding terms until the contribution of the next term drops below this level.

Write a program that gets a (small) double, epsilon, from the user, and approximates π within this function.

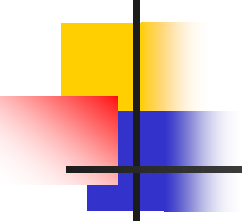


pi (advanced)- solution

```
#include <math.h>
double pi(double epsilon) {
    double n=1, result = 0, term = 1;

    while(sqrt(term*6) >= epsilon) {
        result += term;
        n=n+1;
        term = 1.0/(n*n);
    }

    return sqrt(result*6);
}
```

pi (advanced)- solution cont.

```
/*This program approximates pi*/
```

```
#include <stdio.h>
```

```
int main(void) {  
    double epsilon;
```

```
    printf("enter approximation factor\n");  
    scanf("%lf",&epsilon);
```

```
    printf("The calculated approximation of pi is: %g\n",pi(epsilon));
```

```
    return 0;  
}
```