**Nguyễn Mạnh Bảo Tín MSSV SE170134**

**[tinnmbse170134@fpt.edu.vn](mailto:tinnmbse170134@fpt.edu.vn)**

**Subject: PRF192- PFC**

**Workshop 03 (ODD ONES ONLY)**

**Objectives:**

1. Practicing skills at analyzing and implementing programs using user-defined functions.
2. Making familiar with some basic algorithms

**Grading** **10 programs, 1 mark/program**

**Program 1:**

|  |  |
| --- | --- |
| **Objectives** | Practice implementing simple functions |
| **Related knowledge** | **Definition**: A prime is positive integer that is greater than 1 **and** it is the multiple of 1 and itself only.  **Theorem**: The integer n is a prime if and only if n>1 and it can not be divided by all integers from 2 to ⎣ square root of n⎦ .  Use the library **math.h** to get the function **sqrt(double)** for getting the square root of a positive number. |
| **Problem** | Write a C that will accept a positive integer n, n>=2 then print out primes between 2 and n. |
| **Analysis**  *Nouns:*  *positive integer*  *🡪 int n* | **Suggested algorithm (logical order of verbs)**  Begin  Do {  Accept n;  }  While ( n<2);  For ( i=2 to n )  If (**i is a prime** ) Print out i; 🡪 **Function int prime (int i)**  End |
| **Algorithm for checking whether an integer is a prime or not** | **int prime( int n ) {**  **int m = sqrt(n);** /\* m: square root of n \*/  **int i;**  /\* variable having value from 2 to m \*/  **if (n<2) return 0;** /\* Condition 1 is not satisfied \*/  **for ( i=2; i<=m; i++)** /\* checking the second condition \*/  **if (n%i==0) return 0 ;** /\* n is divided by i 🡪 n is not a prime \*/  **return 1;** /\* n is a prime \*/  **}** |

**Program 1:**

#include <stdio.h>

#include<math.h>

int prime(int n);

int main()

{

int n, i;

printf("Enter n: ");

scanf("%d", &n);

printf("Prime numbers between %d and 2 are: ", n);

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

{

if(prime(i) ==1)

{

printf("%d",i);

}

}

return 0;

}

int prime(int n)

{

if (n<2)

{

return 0;

}

int i;

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

{

if(n%i==0)

{

return 0;

break;

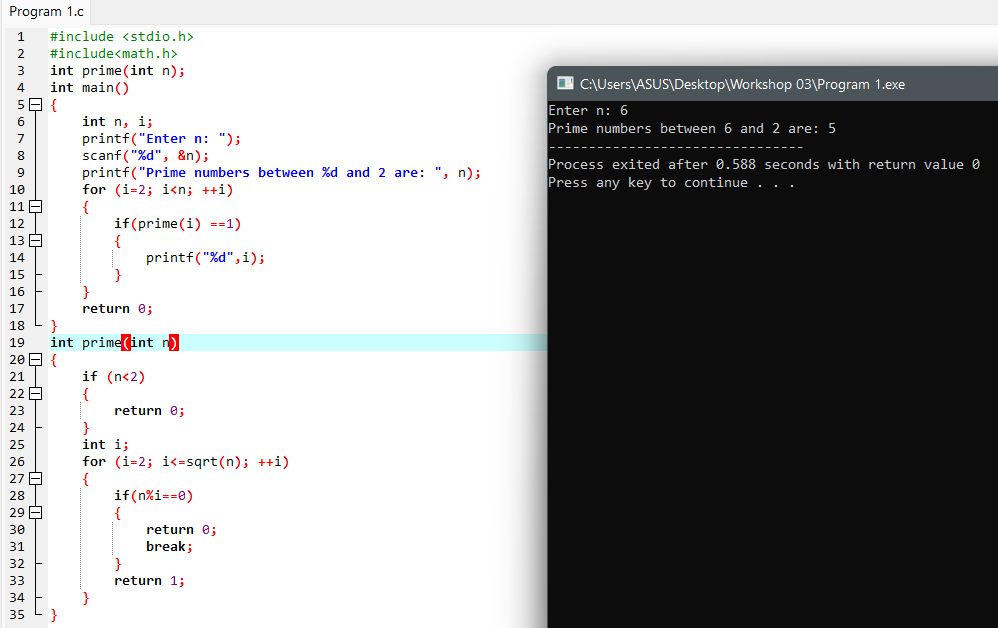
}

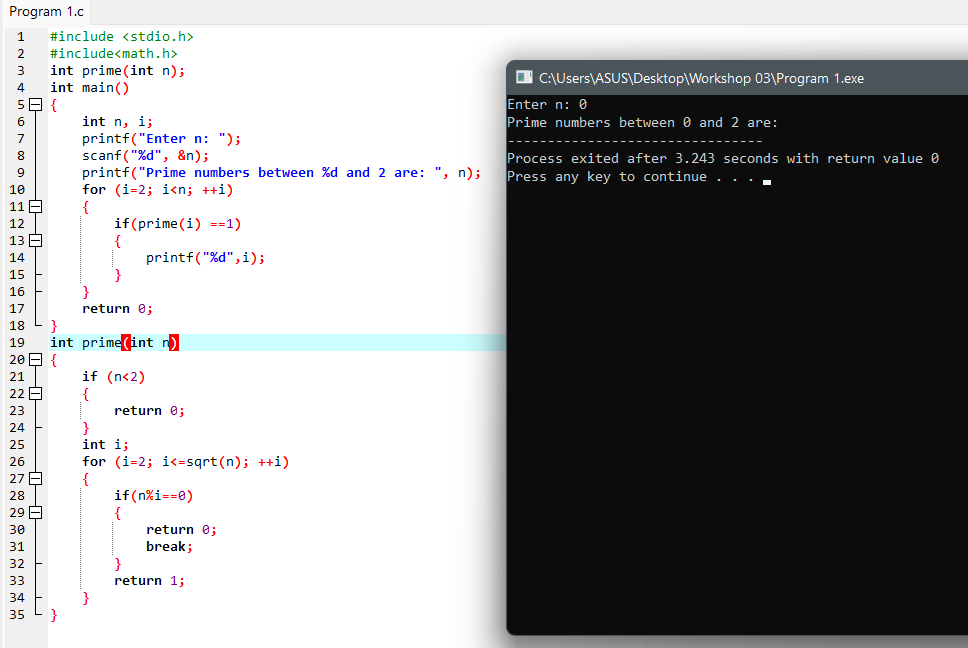
return 1;

}

}

**Run program 1:**





**Program 3:**

|  |  |
| --- | --- |
| **Objectives** | Practice implementing simple functions |
| **Related knowledge** | A point p is in a circle if the distance from the center to p is less than the radius. |
| **Problem** | Write a C program that will accept a point and a circle having the center is (0,0) then print out the relative position of this point with the circle. |
| **Analysis**  Nouns:  A point 🡪 double x,y  A circle 🡪 double r  Relative position   * int result * -1: (x,y) is out of the circle * 0: (x,y) is on the circle * 1: (x,y) is in the circle | **Suggested algorithm (logical order of verbs)**  Begin  Accept x, y;  Do {  Accept r;  }  While(r<0);  result = getRelPos(x,y,r);  if (result ==1) Print out “The point is in the circle”;  else if (result==0) Print out “The point is on the circle”;  else Print out “The point is out of the circle”;  End |
| **Algorithm for**  **getting relative position of a point with a circle** | **int getRelPos ( double x, double y, double r) {**  **double d2=x\*x + y\*y;** /\* d2= x2+ y2 \*/  **double r2= r\*r;** /\* r2\*/  **if (d2<r2) return 1 ;** /\* d2<r2 🡪the point is in the circle \*/  **else if (d2==r2) return 0 ;** /\* d2=r2 🡪the point is on the circle \*/  **return -1 ;** /\* d2 > r2 🡪the point is out of the circle \*/  **}** |

**Program 3:**

#include <stdio.h>

#include<math.h>

int getRelPos(double x, double y, double r);

int main()

{

double x, y ,r, result;

printf("Input point (format as follow: x,y): ");

scanf("%d,%d", &x, &y);

do

{

printf("Input r: ");

scanf("%d", &r);

}

while (r<0);

result = getRelPos (x, y ,r);

if (result =1)

{

printf("The point is inside of the circle");

}

else if (result ==0)

{

printf("The point is on the circle");

}

else

{

printf("The point is outside of the circle");

}

}

int getRelPos(double x, double y, double r)

{

double d2=x\*x + y\*y;

double r2=r\*r;

if(d2<r2) return 1;

else if(d2==r2) return 0;

return -1;

}

**Program 5:**

|  |  |
| --- | --- |
| **Objectives** | Practice implementing simple functions |
| **Related knowledge** | Fibonacci sequence: 1 1 2 3 5 8 13 21 34 …  Two first numbers: 1  Others: Its value is the sum of 2 previous numbers |
| **Problem** | Write a C program that will print out the value at the nth position in Fibonacci sequence. |
| **Analysis**  A position   * int n | **Suggested algorithm (logical order of verbs)**  Begin  Do {  Accept n;  }  While (n<1);  Print out fibo(n);  End. |
| **Algorithm for**  **Computing the nth value of the Fibonacci sequence** | **double fibo ( int n) {**  **int t1=1, t2=1, f=1, i ;**  **for ( i= 3, i<=n; i++) {**  **f= t1 + t2;**  **t1= t2;**  **t2=f;**  **}**  **return f;**  **}** |

**How to compute the nth value of the Fibonacci sequence**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Position 1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1** | **1** | **2** | **3** | **5** | **8** | **13** | **21** | **34** | **55** |
| T1 | T2 | F |  |  |  |  |  |  |  |
|  | T1 | T2 | F |  |  |  |  |  |  |
|  |  | T1 | T2 | F |  |  |  |  |  |
|  |  |  | T1 | T2 | F |  |  |  |  |
|  |  |  |  | T1 | T2 | F |  |  |  |
|  |  |  |  |  | T1 | T2 | F |  |  |
|  |  |  |  |  |  | T1 | T2 | F | … |

**Program 5:**

#include <stdio.h>

double fibo (int n)

{

int t1=1, t2=1, f=1, i;

for (i=3; i<=n; i++)

{

f= t1+t2;

t1=t2;

t2=f;

}

return f;

}

int main()

{

int n;

do

{

printf("Input n: ");

scanf("%d",&n);

}

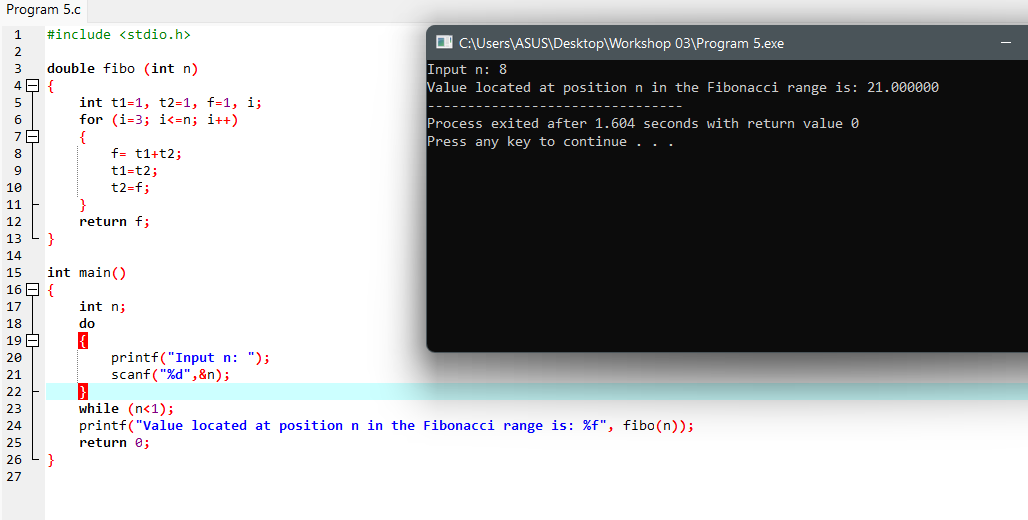
while (n<1);

printf("Value located at position n in the Fibonacci range is: %f", fibo(n));

return 0;

}

Run program 5:



**Program 7:**

|  |  |
| --- | --- |
| **Objectives** | Practice implementing simple functions |
| **Related knowledge** | Getting the rightmost digit of the integer n: **n%10** |
| **Problem** | Write a C program that will carry out some times. In each time, a nonnegative integer is accepted then print out the sum of its decimal digits. The program will terminate when its value of accepted number is negative. |
| **Analysis**  Sum 🡪 int S=0  Accepted integer   * int n | **Suggested algorithm (logical order of verbs)**  Begin  Do  { Accept n;  If (n>=0)  { S = sumDigits(n);  Print out S;  }  }  While (n>=0);  End |
| **Algorithm for**  **Computing sum of digits of a nonnegative integer** | **int sumDigits (int n)**  **{** int sum=0; /\* initialize sum of digits \*/  Do  { int remainder = n%10 ; /\* Get a digit at unit position \*/  n = n/10;  sum += remainder;  }  while (n>0);  return sum;  **}** |

Program 7:

#include <stdio.h>

int sumDigits(int n)

{

int sum=0;

do

{

int remainder = n%10;

n = n/10;

sum +=remainder;

}

while (n>0);

return sum;

}

int main()

{

int n,s;

do

{

printf("Input n: ");

scanf("%d", &n);

if (n>=0)

{

s= sumDigits(n);

printf("Sum= %d\n", s);

}

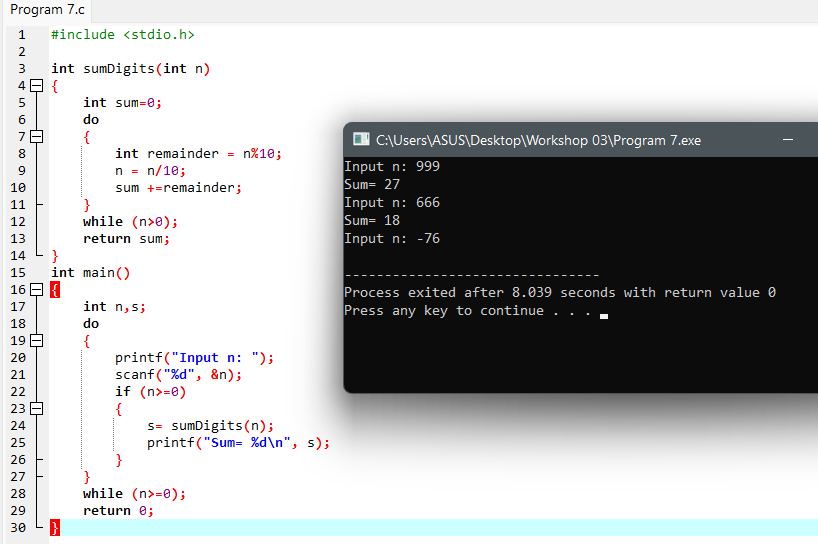
}

while (n>=0);

return 0;

}

Run program 7:



**Program 9:**

|  |  |
| --- | --- |
| **Objectives** | Practice implementing simple functions |
| **Related knowledge** | Find out the greatest common divisor **(gcd)** and least common multiple **(lcm)** of two positive integers:  *Find out gcd of a and b*  a b a b  14 21 13 8  14 7 5 8  7 7 5 3  2 3  2 1  1 1  **int gcd( int a, int b)**  { while ( a != b )  if a>b then a -=b;  else b -= a;  return a;  }  **int lcm ( int a, int b)**  { return a\*b/ gcd(a,b);  } |
| **Problem** | Write a C program that will accept two positive integers then print out their greatest common divisor and least common multiple. |
| **Analysis**  Two integers   * int a, b   gcd 🡪 int d  lcm 🡪 int m | **Suggested algorithm (logical order of verbs)**  Begin  Do  { Accept a, b;  }  While ( a<=0 OR b <=0);  d = gcd(a,b);  m = lcm (a.b);  Print out d;  Print out m;  End |

Program 9:

#include<stdio.h>

int gcd(int a, int b)

{

while (a!=b)

if (a>b)

{

a -=b;

}

else

{

b -= a;

}

return a;

}

int lcm (int a, int b)

{

return a\*b/gcd(a, b);

}

int main()

{

int a, b, d, m;

do

{

printf("Input a: ");

scanf("%d", &a);

printf("Input b: ");

scanf("%d", &b);

}

while (a<=0 || b<=0);

d = gcd(a,b);

m= lcm(a,b);

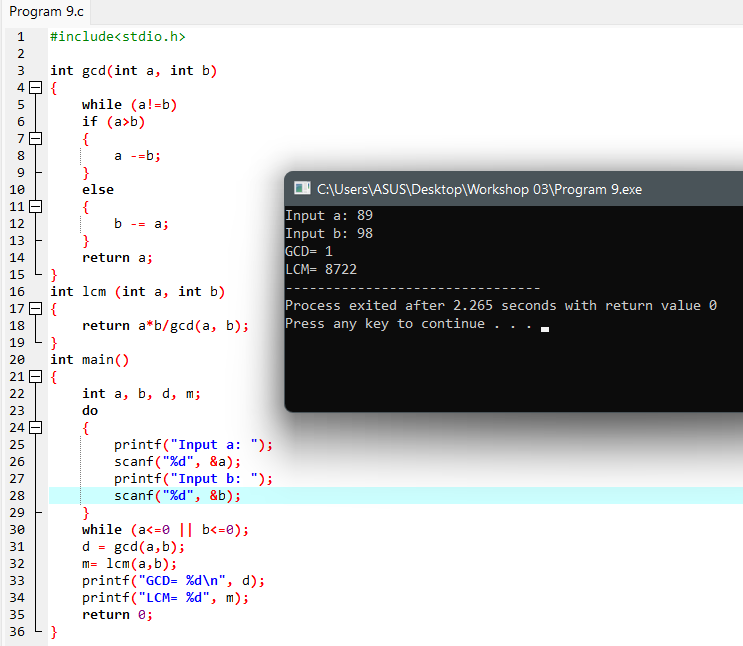
printf("GCD= %d\n", d);

printf("LCM= %d", m);

return 0;

}

Run program 9:



Related files can be found at: [Workshop 3 PRF192](https://fptuniversity-my.sharepoint.com/:f:/g/personal/tinnmbse170134_fpt_edu_vn/Ehvm5fZcV0FArYW5dLVuDasBR9xo4GjX_rnJ98bri3ZK3Q?e=9ElWwp)