C assignment 2. Parts A and B.

03/30/2019
Armandas Rokas (s185144)
Github readme link: https://github.com/ArmandasRokas/C_assigment_2/blob/master/docs/readme.md
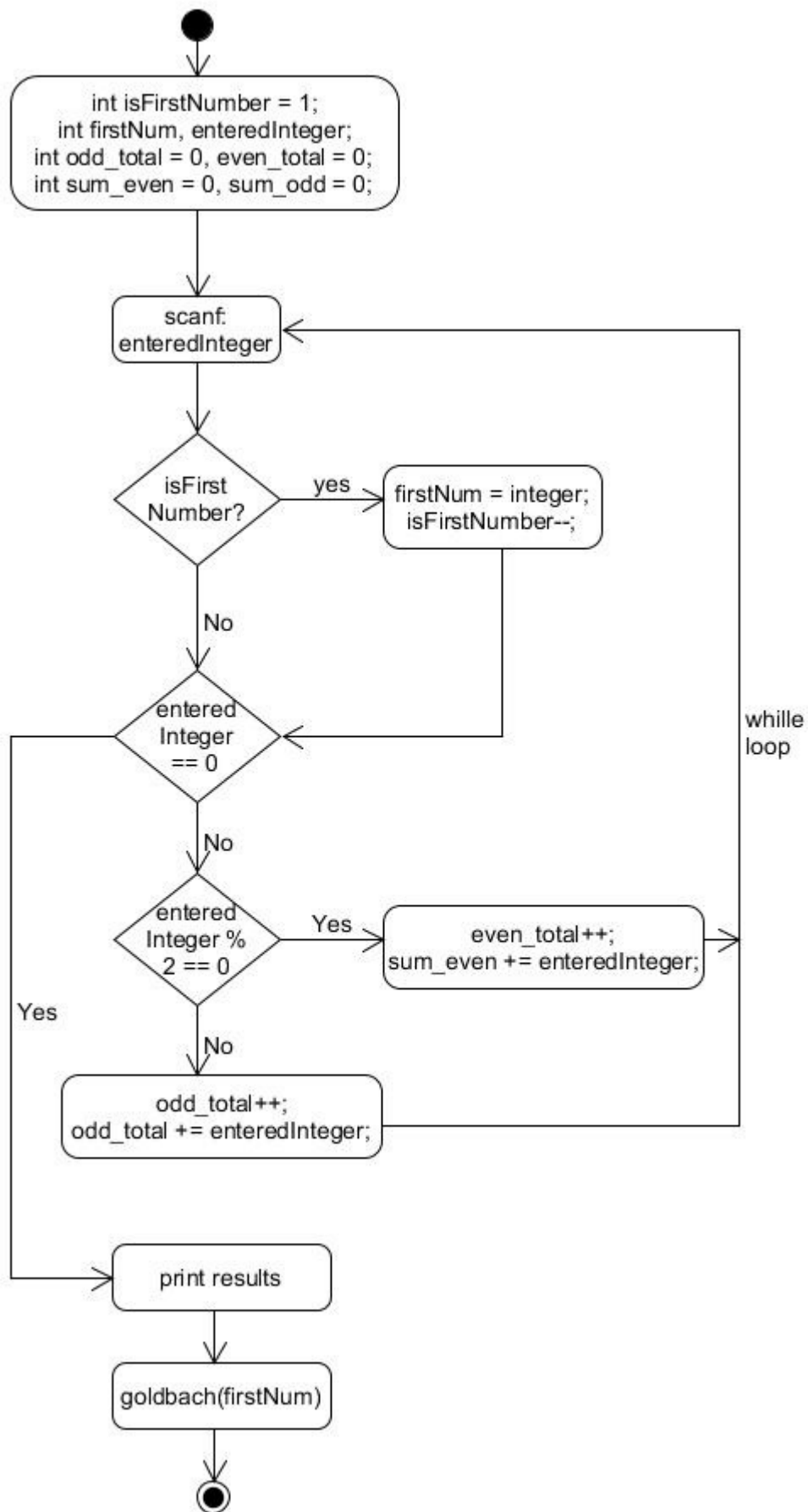
Even_odd function

Firstly, the function has counters for the total sum of even numbers and odd numbers. Secondly there are also two more counters, which holds the values of how many times odd number and even number was entered.
When user exits decides to terminate the progam it divides the sum of even/odd numbers by the number of times entered even/odd numbers and prints results.

Difficulties

There was a bug which was caused by division by 0, when no even or odd numbers was added. The bug is fixed by adding extra selection statments at the end of program, which checks if the number of times entered even/add numbers is not equal to 0.

The flow diagram of even_odd function

```
                    ●
                    │
                    ▼
        ┌───────────────────────────────┐
        │   int isFirstNumber = 1;       │
        │   int firstNum, enteredInteger;│
        │ int odd_total = 0, even_total = 0;│
        │ int sum_even = 0, sum_odd = 0;  │
        └───────────────────────────────┘
                    │
                    ▼
              ┌───────────┐
              │  scanf:    │◄──────────────────────────┐
              │enteredInteger│                          │
              └───────────┘                            │
                    │                                  │
                    ▼                                  │
                  ◇ isFirst    yes   ┌──────────────────┐│
                  ◇ Number? ───────► │ firstNum = integer;││
                  ◇               │  │ isFirstNumber--;  ││
                    │                └──────────────────┘│
                   No                                    │
                    │                                    │
                    ▼                                    │
                  ◇ entered                      whille   │
                  ◇ Integer  ◄─────────────────   loop    │
                  ◇  == 0                                 │
                    │                                     │
                   No                                     │
                    │                                     │
                    ▼                                     │
        ┌── ◇ entered    Yes   ┌──────────────────────┐   │
        │   ◇ Integer % ─────► │ even_total++;         │──►┤
   Yes  │   ◇  2 == 0          │ sum_even += enteredInteger;│
        │     │                └──────────────────────┘   │
        │    No                                           │
        │     │                                           │
        │     ▼                                           │
        │  ┌────────────────────────┐                     │
        │  │ odd_total++;            │─────────────────────┘
        │  │ odd_total += enteredInteger;│
        │  └────────────────────────┘
        │
        ▼
   ┌──────────────┐
   │ print results│
   └──────────────┘
          │
          ▼
   ┌──────────────────┐
   │ goldbach(firstNum)│
   └──────────────────┘
          │
          ▼
          ◉
```

Goldbach function

The goldbanch fuction prints all natural numbers less tha n, w hich cannot be represented as a sum of tw o prime numbers.
The function itself just iterates n times in checks, if the n can be represnted as a sum of tw o prime numbers by taking advantage
of another function, w hich its explain below.

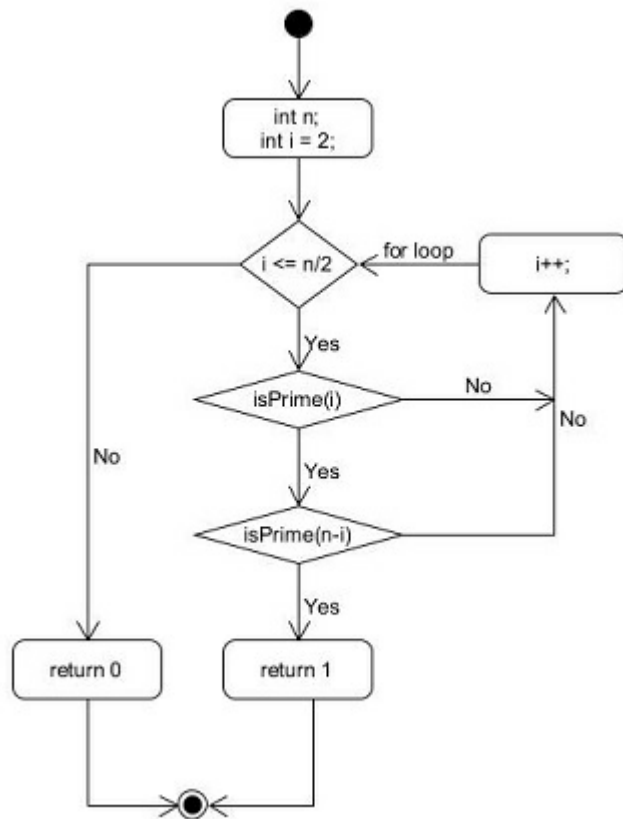## The flow diagram of goldbach function
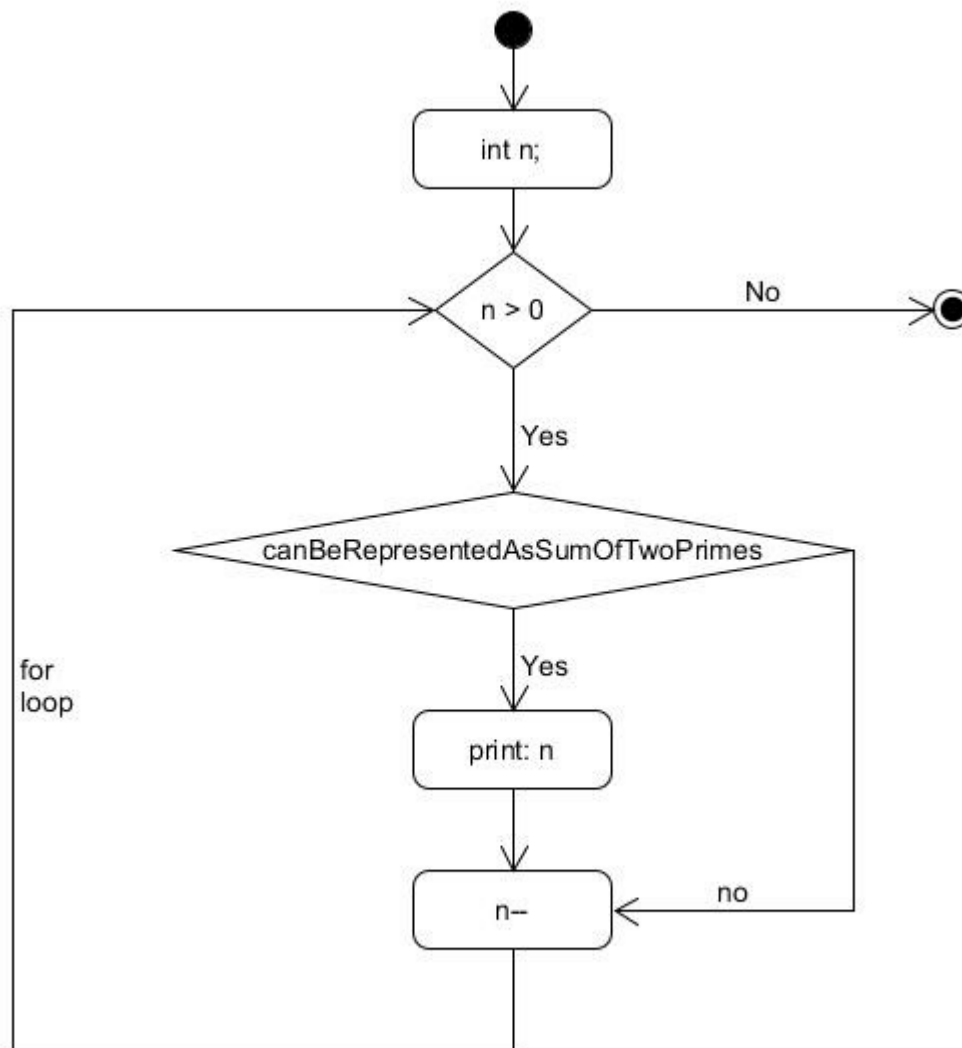
CanBeRepresentedAsSumOfTwoPrimes function

The function takes a parameter n and checks n can be presented by sum as sum of tw o primes. The function iterates at maximum
n/2 times. It tries to find a prime numbers w hich is less than half of n. Than it subtracts from n and checks if the result is prime
number. If so that means function can be represented as sum of tw o primes and returns 1, w hich means 'true'.

The flow diagram of canBeRepresentedAsSumOfTwoPrimes function

IsPrime function

Again the function takes n as paramater, iterates n/2 and uses modulus to defines if the number can be devided by a current i and get a whole number. If not a such case exists, so the number is prime.

The flow diagram of isPrime function

```
     ●
     │
     ▼
┌──────────────┐
│   int n;     │
│   int i = 2; │
└──────────────┘
     │
     ▼
    ◇ i <= n/2 ◇ ◀─── for loop ─── ┌──────────┐
     │         │                    │   i++;   │
     │ No      │ Yes                └──────────┘
     │         │                          ▲
     │         ▼                          │ No
     │      ◇ isPrime(i) ◇ ──── No ───────┤
     │         │                          │ No
     │         │ Yes                      │
     │         ▼                          │
     │      ◇ isPrime(n-i) ◇ ─────────────┘
     │         │
     │         │ Yes
     │         ▼
┌──────────┐ ┌──────────┐
│ return 0 │ │ return 1 │
└──────────┘ └──────────┘
     │            │
     └──────►◉◄───┘
```