GENESI DEI NUMERI PRIMI

* Legge di Genesi e Struttura dei Numeri Primi

* Algoritmo detto “Metodo del compasso e della gittata numerica”

* Implementazione rudimentale dell'algoritmo detto “metodo del compasso e della gittata numerica” in linguaggio C/C++

ISMAELE ALONGI

Note sulla pubblicazione

Autore: ***Ismaele Alongi (nato a Caltanissetta nel 1977 in Italia)***
Nazionalità dell'autore: ***ITALIANA***
Luogo di pubblicazione: ***Sommatino (CL)***
Pubblicatore e stampatore: ***Ismaele Alongi – Via Angelo Alaimo, 21***
C.A.P. 93019 – Sommatino (CL)
ITALIA / ITALY
Opera pubblicata sui siti Internet: ***genesi-numeri-primi.ismael.it***
genesis-of-prime-numbers.ismael.it
Anno di pubblicazione: ***2006***
e-mail dell'autore: ***infoarte@libero.it***

GENESI DEI NUMERI PRIMI

Legge di Genesi e Struttura dei Numeri Primi

"La bellezza, la sapienza, la potenza e la gloria appartengono a Dio, l'Altissimo, Colui che fa divenire.

Egli ci dà il privilegio e la gioia di poter conoscere il Suo eterno proposito per l'uomo."

Ismaele Alongi.

Da un' analisi dei numeri primi ho dedotto la seguente Legge di Genesi e Struttura dei Numeri Primi:

Nella successione numerica dei numeri primi, ogni nuovo numero primo nasce nel primo varco numerico lasciato vuoto dalle progressioni numeriche dei precedenti numeri primi noti.

Originariamente avevo espresso questo concetto così:

Ogni nuovo numero primo viene generato nel primo spazio lasciato vuoto dalle serie numeriche inferiori.

La successione numerica dei numeri primi giace su una semiretta che ha origine nel campo del finito e punta all'infinito. Idealmente ho immaginato delle successioni numeriche sovrapposte che hanno origine da punti situati su una retta immaginaria ortogonale alla semiretta dei numeri naturali e passante per il punto zero. E' straordinario immaginare tutti i numeri primi allineati su questa retta immaginaria: virtualmente immaginiamo un immenso e infinito numero composto che va a colmare l'infinito... o potrebbe anche essere l'infinito numero primo... Comunque sia, ci sono interrogativi che per l'uomo sono imperscrutabili, proprio perché la mente umana opera nel campo del finito e, dato che l'infinito è incolmabile, cercare di chiudere il cerchio infinito porta in errore.

L'idea dei nuovi spazi vuoti che vengono via via colmati dai nuovi numeri primi, scaturisce dall'osservazione dello speciale numero primo pari: il numero due.

Se infatti osserviamo il comportamento della progressione del numero due, notiamo subito che va a colmare ½ infinito dei numeri naturali. Cosa sarebbe necessario per colmare il resto dell'infinito? Una progressione numerica in base uno con ragione due. Non è questa però la logica della successione dei numeri primi. L'infinito viene colmato gradualmente, e dato che l'infinito non si può colmare, non si colmerà mai, quindi per la mente umana esistono infiniti numeri primi. Resterà sempre infatti un primo varco numerico vuoto oltre le progressioni sovrapposte dei numeri primi noti: quel primo varco numerico vuoto è uno “zero”, per niente banale, come amava dire Bernhard Riemann, e quel vuoto numerico è occupato da un nuovo numero primo da conoscere, un nuovo pianeta numerico con una sua precisa traiettoria ciclica, una frequenza con una precisa lunghezza d'onda.

Non ci vuole tanto per azzardare l'idea che esplorando i numeri primi lungo la semiretta dei numeri naturali, stiamo esplorando solo metà dell'infinito, l'altra metà si trova allo specchio, sulla semiretta virtuale dei numeri negativi, quindi molto probabilmente siamo di fronte a un miraggio matematico se pensiamo di vedere sull'allineamento virtuale che passa per lo zero-origine un unico e infinito numero composto, semmai nel miraggio c'è un infinito numero primo.

Non so se queste mie ipotesi vanno a dimostrare l'ipotesi di Riemann, che mi sembra quasi un azzardo matematico riuscito, ma questo risultato che ho ottenuto nasce da un'altra ricerca che sto ancora sviluppando: la fattorizzazione dei numeri composti.

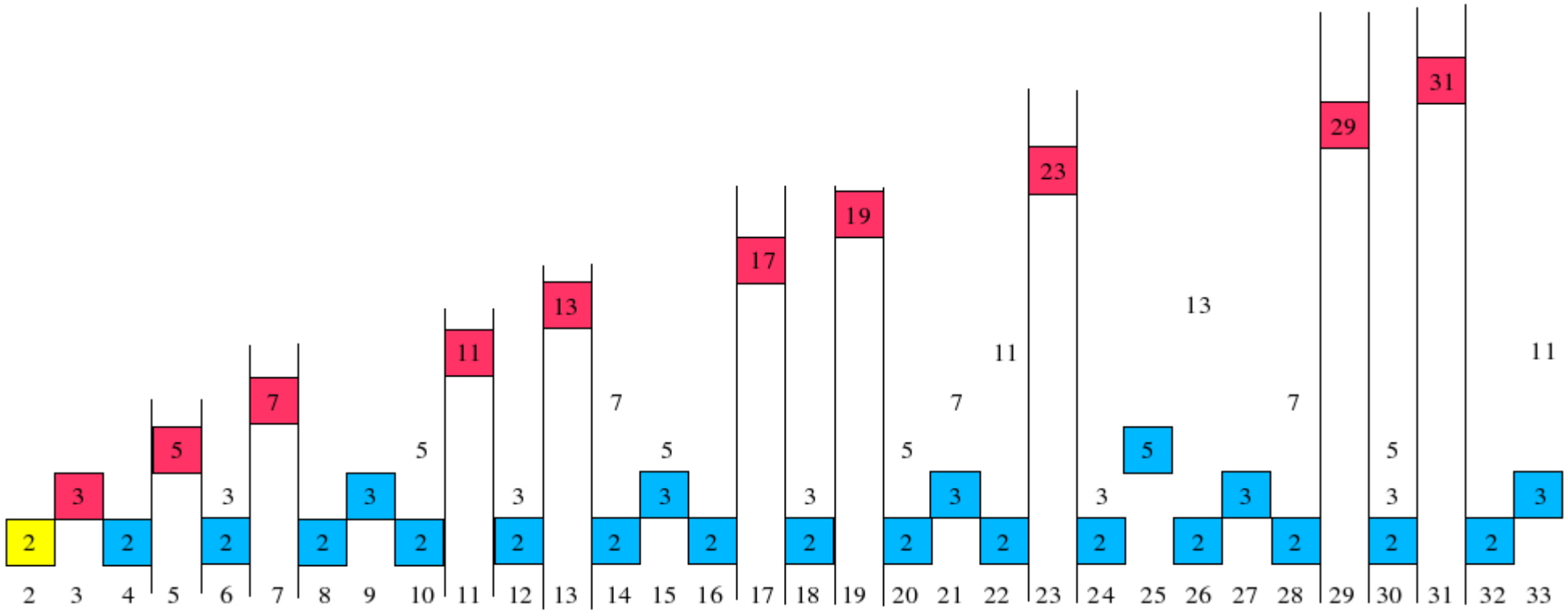
Dalla legge di successione numerica e genesi dei numeri primi che ho dedotto, ho preso spunto per sviluppare un sistema abbastanza efficace per calcolare i successivi numeri primi in un dato intervallo numerico, dati i precedenti numeri primi noti. Questo algoritmo l'ho denominato metodo del compasso numerico e della gittata numerica.

Basterebbe come elemento di base il fondamentale numero due, per ottenere tutti gli altri numeri primi, infatti se osserviamo la progressione del numero due, il primo spazio numerico vuoto è tra il due e il quattro, quindi il prossimo numero primo è il tre. Osserviamo poi le progressioni sovrapposte del due e del tre: il prossimo varco numerico è il cinque ecc.

Per visualizzare meglio il concetto ho realizzato un semplice grafico. La mia scoperta infatti è nata nel momento in cui ho iniziato a considerare per i vari numeri naturali che ho analizzato in successione, il minimo divisore maggiore dell'unità relativamente a ciascun numero, quindi ho iniziato a intravedere delle simmetrie che si ripetevano ciclicamente, quasi come una fantastica sequenza genetica numerica: il DNA dei numeri primi.

Legge di Genesi e Struttura dei Numeri Primi

Rappresentazione grafica



GENESI DEI NUMERI PRIMI

Algoritmo detto “Metodo del compasso e della gittata numerica”

Dati iniziali
Sequenza di numeri primi noti.
Conosciamo la sequenza dei numeri primi da 2 a “n”.

Fase di calcolo 0
Centro di simmetria e raggio di simmetria
Le progressioni generate dai vari numeri primi da 2 a “n” creano delle “simmetrie”, quindi decidiamo di considerare la posizione dell'ultimo numero primo conosciuto “n” come “centro della simmetria”, ovvero come centro della semicirconferenza tracciata avente come centro il punto “n”. Il valore dell'ultimo numero primo conosciuto lo considereremo anche come “raggio di simmetria”.

Elementi della progressione del numero 2 dentro il raggio di simmetria.
Quanti elementi avrà la progressione base 2 e ragione 2 all'interno del raggio di simmetria? Il numero di elementi (di seguito nominato “n_due”) sarà pari a: (“raggio_di_simmetria” – 1) / 2.
A quanto ammonterà il valore dell'ultimo elemento della progressione del 2 (di seguito nominato “ultimo_due”) all'interno del raggio di simmetria?
“ultimo_due” = 2 + (“n_due” – 1) * 2 .
Questo calcolo si basa sulla formula per le progressioni aritmetiche:

$$a_n = a_1 + (n - 1) * d;$$

dove a_n è l'ultimo elemento della progressione, a_1 è il primo elemento della progressione, n è il numero di elementi della progressione, d è la ragione.

Fase di calcolo 1
Moduli delle traiettorie incompiute
Per ogni numero primo conosciuto calcolare il modulo (o resto) della divisione tra “centro_di_simmetria” e il numero primo esaminato, quindi memorizzare il modulo che rappresenta, per il relativo numero primo a esso associato, la sua traiettoria incompiuta all'interno del raggio di simmetria.

Fase di calcolo 2
Traiettorie mancanti
Per ciascun numero primo calcolare la differenza tra il numero primo esaminato e il relativo modulo: questa differenza è la traiettoria mancante del relativo numero primo esaminato all'interno della “gittata numerica” oltre l'ultimo numero primo conosciuto, comunque prima di 2n .

Fase di calcolo 3
Creazione delle progressioni dei numeri primi dispari inferiori a "n" all'interno della "gittata numerica" in base alle traiettorie mancanti

Per ciascun numero primo calcolare gli elementi della propria progressione all'interno della gittata numerica (che è pari al raggio di simmetria) in base alle traiettorie mancanti e in base alle seguenti regole:

- se la traiettoria mancante è pari, la base della progressione equivale alla traiettoria mancante;
- se la traiettoria mancante è dispari, la base della progressione equivale alla traiettoria mancante + il relativo numero primo;
- calcolare la progressione fino a che l'ultimo elemento della progressione è minore o uguale all'ultimo elemento della progressione del due che abbiamo calcolato in precedenza ("ultimo_due"), secondo la seguente regola di calcolo che ci permette di saltare i numeri composti pari all'interno della gittata numerica: $k_y = k_x + 2 * p$; dove k_y rappresenta il successivo elemento della progressione numerica esaminata, k_x il precedente elemento della progressione in questione, p rappresenta il numero primo relativo alla progressione che stiamo esaminando;
- memorizziamo gli elementi delle varie progressioni calcolate all'interno della "gittata numerica": queste sono le intersezioni con la semiretta dei numeri naturali.

Fase di calcolo 4
Calcolo degli zeri, posizioni dei nuovi numeri primi

Calcoliamo ogni elemento della progressione numerica del 2 all'interno della gittata numerica fino al limite di simmetria, in base zero e ragione 2, quindi in progressione verifichiamo l'esistenza di intersezioni con la semiretta dei numeri naturali: se l'elemento considerato esiste nelle precedenti progressioni numeriche, vuol dire che esiste già l'intersezione con la semiretta dei numeri naturali, quindi siamo in presenza di un numero composto; se invece l'elemento della progressione del due - che stiamo esaminando - non esiste nelle precedenti progressioni, vuol dire che siamo in presenza di un vuoto numerico, uno spazio vuoto ovvero di uno "zero" non banale, come direbbe Riemann. La somma della posizione dello zero + il numero primo "n" (ovvero l'ultimo numero primo conosciuto che abbiamo nominato anche "centro di simmetria") dà come risultato il successivo numero primo nella successione dei numeri primi. Continuare con i calcoli fino al limite di simmetria, perché molto probabilmente troveremo ulteriori numeri primi all'interno della gittata numerica.

Note
Nel caso in cui non dovessero essere trovati nuovi numeri primi all'interno della gittata numerica (la quale è pari al raggio di simmetria), procedere con una nuova gittata numerica. In base al postulato di Bertrand, però, è alquanto ragionevole ipotizzare che all'interno della gittata numerica pari al raggio di simmetria (ovvero fino alla posizione 2n) troveremo sempre dei numeri primi.

Ho eseguito una rudimentale implementazione di questo algoritmo di calcolo in linguaggio C/C++ ed è possibile eseguire il programma per verificare in parte le funzionalità di questo metodo di calcolo, o meglio per chiarire le idee su come vanno formulati effettivamente i calcoli.

Esempio di applicazione dell'algoritmo denominato "metodo del compasso e della gittata numerica".

Data la successione dei numeri primi da 2 a 19.

Fase di calcolo 0
Centro di simmetria e raggio di simmetria.

```
n = 19;     // n = elemento[8];
centro_simmetria = 19;   // centro_simmetria = n;
```

Elementi della progressione del numero 2 dentro il raggio di simmetria.

```
n_due = (19 – 1) / 2;              //  n_due = (n – 1) / 2;
```

Fase di calcolo 1
Moduli delle traiettorie incompiute

```
// modulo[i] = n % elemento[i];

modulo[2] = 19 %  3;      // modulo[2] = 1;
modulo[3] = 19 %  5;      // modulo[3] = 4;
modulo[4] = 19 %  7;      // modulo[4] = 5;
modulo[5] = 19 % 11;      // modulo[5] = 8;
modulo[6] = 19 % 13;      // modulo[6] = 6;
modulo[7] = 19 % 17;      // modulo[7] = 2;
```

Fase di calcolo 2
Traiettorie mancanti

```
// tm[i] = elemento[i] - modulo[i];

tm[2] =  3 - modulo[2];    // tm[2] =  3 – 1;   -->   2
tm[3] =  5 - modulo[3];    // tm[3] =  5 – 4;   -->   1
tm[4] =  7 - modulo[4];    // tm[4] =  7 – 5;   -->   2
tm[5] = 11 - modulo[5];    // tm[5] = 11 – 8;   -->   3
tm[6] = 13 - modulo[6];    // tm[6] = 13 – 6;   -->   7
tm[7] = 17 - modulo[7];    // tm[7] = 17 – 2;   --> 15
```

Fase di calcolo 3
Creazione delle progressioni dei numeri primi dispari inferiori a "n" all'interno della "gittata numerica" in base alle traiettorie mancanti

- progressione del numero 3, in base alla traiettoria mancante:
  base = 2; // tm[i] è pari, quindi: base=tm[i];
  ragione = 2 * elemento[i];

  quindi otteniamo 2, 8, 14, ~~20;~~
- progressione del numero 5, in base alla traiettoria mancante:
  base = 1 + elemento [i]; // tm[i] è dispari, quindi: base=tm[i]+elemento[i];
  ragione = 2 * elemento[i];

  quindi otteniamo 6, 16, ~~26;~~

- progressione del numero 7, in base alla traiettoria mancante:
  2, 16, ~~30;~~

- progressione del numero 11, in base alla traiettoria mancante:
  14, ~~36~~ ;

- progressione del numero 13, in base alla traiettoria mancante:
  ~~20~~;

- progressione del numero 17, in base alla traiettoria mancante:
  ~~32~~ ;

Fase di calcolo 4
Calcolo degli zeri, posizioni dei nuovi numeri primi

In base alla fase di calcolo 3, abbiamo notato che esistono le seguenti intersezioni con la semiretta dei numeri naturali, all'interno della gittata numerica:
2,8,14;
6,16;
2,16;
14.

La progressione in base zero e ragione due fino al limite di simmetria n, produce 9 elementi:
2, 4, 6, 8, 10, 12, 14, 16, 18.

I seguenti numeri non esistono tra le intersezioni, sono i primi spazi vuoti o varchi numerici denominati zeri, ovvero le posizioni dei nuovi numeri primi dentro la gittata numerica: 4, 10, 12, 18.

La formula quindi è semplice: zero = n + posizione;

// elemento[i] = n + posizione;

elemento[9] = n + 4; // 23 = 19 + 4;
elemento[10] = n + 10; // 29 = 19 + 10;
elemento[11] = n + 12; // 31 = 19 + 12;
elemento[12] = n + 18; // 37 = 19 + 18;

ATTENZIONE !!! LEGGERE CON CURA LE SEGUENTI CONDIZIONI DI UTILIZZO PRIMA DI PROSEGUIRE NELLA LETTURA DEL DOCUMENTO.

Accordo di Licenza d'uso a titolo non esclusivo tra il proprietario della proprietà intellettuale e l'utente finale.

Il proprietario intellettuale è Ismaele Alongi (di seguito indicato come “l'autore”), nato nel 1977 in Italia, autore dell'opera oggetto della proprietà intellettuale dal titolo “Genesi dei Numeri Primi - Implementazione rudimentale dell'algoritmo detto “metodo del compasso e della gittata numerica” in linguaggio C/C++.

La suddetta opera oggetto della proprietà intellettuale è costituita dal programma per computer e l'algoritmo in esso espresso e rappresentato, entrambi ideati, creati e scritti dall'autore Ismaele Alongi (di seguito indicata come “il seguente programma per computer e l'algoritmo in esso rappresentato”), la quale opera si trova nelle pagine seguenti di questo documento.

L'utente finale è chiunque decide di leggere, utilizzare o impiegare il seguente programma per computer e l'algoritmo in esso rappresentato, accettando senza riserve le condizioni di questo accordo di licenza d'uso.

Il seguente programma per computer e l'algoritmo in esso rappresentato sono concessi in licenza d'uso a titolo non esclusivo “così per come sono” e senza alcuna garanzia implicita o esplicita, per un solo utente e per un solo computer. Il seguente programma per computer e l'algoritmo in esso rappresentato non sono freeware, bensì vengono semplicemente mostrati al pubblico per meglio esprimere e rendere meglio comprensibili le idee dell'autore al fine di comunicarle al pubblico.

Chiunque non è d'accordo con le condizioni di questo accordo di licenza non è autorizzato a leggere e/o utilizzare in alcun modo il seguente programma per computer e l'algoritmo in esso rappresentato.

Chi invece è d'accordo con le condizioni del presente accordo di licenza, concorda con l'autore in quanto l'utente finale non chiederà alcun tipo di rimborso per danni diretti o indiretti per l'uso proprio e/o improprio di quanto contenuto e/o eseguito dal seguente programma per computer e dall'algoritmo in esso rappresentato e per quanto in esso espresso e/o contenuto, né l'utente finale deve recare danni ad altri soggetti mediante l'uso del seguente programma per computer e dell'algoritmo in esso rappresentato. L'utente finale che è d'accordo con il presente accordo di licenza, il quale utente decide di leggere e/o eseguire e/o utilizzare e/o impiegare in pubblico il seguente programma per computer e l'algoritmo in esso rappresentato, deve comunicare alle persone che direttamente o indirettamente fruiranno dell'applicazione che ne deriva, le condizioni del presente accordo di licenza e mettere tali persone nelle condizioni di decidere individualmente e di esprimere la propria decisione, al fine di accettare o rifiutare le condizioni del presente accordo di licenza, quindi ne conseguirà l'utilizzo autorizzato o meno, in base alle condizioni espresse nel presente accordo di licenza.

L'autore si riserva qualsiasi forma di utilizzazione economica della propria proprietà intellettuale (in special modo per l'algoritmo) e l'utente finale che è d'accordo con il presente accordo di licenza acconsente esplicitamente a questa limitazione d'utilizzo.

La lettura e/o l'uso e/o l'impiego in qualsiasi forma e con qualsiasi mezzo del seguente programma per computer e dall'algoritmo in esso rappresentato, comportano da parte dell'utente finale l'accettazione esplicita e senza riserve delle condizioni espresse nel presente accordo di licenza d'uso a titolo non esclusivo tra il proprietario della proprietà intellettuale e l'utente finale.

GENESI DEI NUMERI PRIMI

Implementazione rudimentale dell'algoritmo detto “metodo del compasso e della gittata numerica” in linguaggio C/C++


```
// Copyright 2006 by Ismaele Alongi. All rights reserved.
// Tutti i diritti riservati in Tutti i Paesi.

// "La bellezza, la sapienza, la potenza e la gloria appartengono a Dio,
//  l'Altissimo, Colui che fa divenire. Egli ci dà la gioia e il
//  privilegio di poter conoscere il Suo eterno proposito per l'uomo."
//                                                    Ismaele Alongi.

// Una prima rudimentale implementazione di una verifica della "Legge di Genesi e
// Struttura dei Numeri Primi", ipotizzata da Ismaele Alongi, e dell'algoritmo
// del “compasso e della gittata numerica” dell'autore Ismaele Alongi.
// Altri nomi da proporre sono: "Legge di successione numerica dei numeri primi"
// oppure "ordine di progressione numerica dei numeri primi"

//-------------------------------------------------------------------------
// Note sul funzionamento del programma: il programma usa come input un
// file di testo con i numeri primi iniziali, quindi genera come output
// un file di testo con i numeri primi individuati nell'ordine successivo
// a quelli iniziali.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <limits.h>
#include <values.h>
#include <float.h>
#include <iostream>
#include <fstream>

#define FALSE 0
#define TRUE 1

using namespace std;

void inizializza_liste(int elementi[]);
void verifica_memorizza_esistenza(int ex, int esistenza[]);
void compasso_numerico(int elementi[]);

main ()
{
int primi_iniziali[1000]; // array dei numeri primi iniziali

inizializza_liste(primi_iniziali);// inizializzazione lista numeri primi

compasso_numerico(primi_iniziali);
                          // funzione di calcolo dei nuovi numeri primi:
                          // implementazione rudimentale, vedi dettagli.

} // fine main program
```

```
//---------------------------------------------------------------------------
void compasso_numerico(int elementi[])
{
// dichiarazione variabili
FILE *destinazione;

int moduli[1000], // array dei moduli
              tm[1000], // array delle traiettorie mancanti
              esistenza[1000]; // array delle intersezioni

int centro_simmetria, i, j,
              base, k, n,
              n_due, ultimo_due,
              posizione=0, zero;

short intersezione=FALSE;

esistenza[0]=0;

destinazione=fopen("nuovi_primi.txt", "w");

// fase calcolo 0: centro di simmetria
n=elementi[0]; // ennesimo primo conosciuto
centro_simmetria=elementi[n];// è l'ultimo numero primo conosciuto

cout << "Eseguito calcolo 0: centro di simmetria\n";

//elementi della progressione del numero due dentro il raggio di simmetria
n_due=(centro_simmetria - 1) / 2; // numero di elementi della
                                  // progressione del due

ultimo_due= 2 + (n_due - 1) * 2; // formula: An = A1 + (n - 1) * d;
                                 // dove An è il termine ennesimo cercato,
                                 // A1 è il primo elemento della progressione,
                                 // n è il numero di elementi della progr.,
                                 // d è la ragione.
cout << "Eseguito calcolo elementi della progressione del numero due dentro "
     <<"il raggio di simmetria.\n";

for (i=2; i<=n; i++)
    cout << "Numeri primi conosciuti, "<< i << " = " << elementi[i] <<"\n";
    cout << "altro "<< elementi[20]<< "altro " << elementi[0]<<"\n";

for (i=2; i<=n; i++)
    {
      // fase calcolo 1: moduli delle traiettorie incompiute
      moduli[i]=centro_simmetria%elementi[i];//esegue il modulo e memorizza
      cout << "Eseguito calcolo 1: moduli delle traiettorie incompiute\n"
           << "moduli[" << i << "] = " << moduli[i] << "\n";

      // fase calcolo 2: traiettorie mancanti
      tm[i]=elementi[i]-moduli[i];
                              // calcola le traiettorie mancanti e memorizza

      cout << "Eseguito calcolo 2: traiettorie mancanti\n"
           << "tm[" << i << "] = " << tm[i] << "\n";
```

```
      // fase calcolo 3: creazione delle progressioni numeriche
      // progressioni dei restanti numeri primi,
      // in base alle traiettorie mancanti

      if (tm[i] % 2 == 0)
         {base=tm[i];}
         // caso pari: base progressione = traiettoria mancante

         else base = tm[i] + elementi[i];
         // caso dispari:
         // base progr. = traiettoria mancante + n. primo relativo

      k=base;

      cout << "base = " << base << "\n";

      if (k == ultimo_due)
          verifica_memorizza_esistenza(k, esistenza);

      if (k < ultimo_due)
         { verifica_memorizza_esistenza(k, esistenza);

           while (k < ultimo_due)
                 { k = k + 2 * elementi[i]; // saltiamo le coincidenze con i
                                            // numeri composti pari
                   if (k<=ultimo_due)
                        verifica_memorizza_esistenza(k, esistenza);
                 }; // fine while
         }; // fine if (k< ultimo_due)

      cout << "Eseguito calcolo 3: creazione delle progressioni numeriche\n"
           << "progressioni dei restanti numeri primi, in base alle "
           << "traiettorie mancanti \n";

      cout << "indice del for " << i << "\n";

    }; // fine for delle fasi di calcolo 1, 2, 3

// fase 4: calcolo degli zeri, posizioni dei nuovi numeri primi
// - calcolare la progressione numerica del 2 fino al limite di simmetria
// - verificare l'esistenza degli elementi della progressione numerica:
//    -- se il numero non esiste, questo è lo "zero", quindi eseguire la
//       somma: centro_simmetria + posizione;
// - il ciclo termina quando si arriva al limite di simmetria
// N.B.: bisogna continuare a raffinare il programma per quando i nuovi
// numeri primi saranno più rari.... perché non so se all'interno della
// gittata della simmetria ci sono sempre nuovi numeri primi, anche se
// si potrebbe azzardare l'ipotesi che ci siano sempre.

for (i=1; i<=n_due; i++)
    {
     posizione=posizione + 2; // stiamo calcolando gli elementi della
                              // progressione del due, uno alla volta

     for (j=1; j<=esistenza[0]; j++)
         if (posizione==esistenza[j])
             { intersezione=TRUE;
               j=esistenza[0];
             };

     if (intersezione==FALSE)  // una volta chiuso il for annidato calcola
                               // lo zero se c'è la condizione di vuoto
        { zero=centro_simmetria + posizione;
          fprintf (destinazione,"%ld\n", zero);
        }
       else intersezione=FALSE;
```

```
      cout << "Eseguito calcolo 4: calcolo degli zeri, posizioni dei nuovi "
           << "numeri primi\n...indice del for "<< i <<"\n";

     }; // fine del "for" della progressione del due "a colmare"
        // dentro il raggio di simmetria

// N.B.: Se non vengono trovati nuovi numeri primi all'interno del raggio
// di simmetria, continuare la ricerca con una nuova "gittata numerica",
// fino a che non li troviamo. Questa opzione è da implementare.

fclose(destinazione);
} // fine funzione compasso_numerico

//---------------------------------------------------------------------------
void inizializza_liste(int elementi[])
{
FILE *origine;

int x=0;

origine=fopen("primi.txt", "r");

while (fscanf(origine, "%ld", &elementi[x+1]) !=EOF)
      x++;

fclose(origine);

elementi[0]=x; // indice dell'array

} // fine procedura inizializza_liste

//---------------------------------------------------------------------------
void verifica_memorizza_esistenza(int ex, int esistenza[])
{
short duplicato=FALSE;
int w;

if (esistenza[0]==0)
   { esistenza[1]=ex;
     esistenza[0]=1;
   }
   else
        { for (w=1; w<=esistenza[0]; w++)
             if (esistenza[w]==ex)
                { duplicato=TRUE;
                  w=esistenza[0];
                }; // fine for--if

             if (duplicato==FALSE)
                { esistenza[0]++;
                  w=esistenza[0];
                  esistenza[w]=ex;
                };
        }; // fine else
} // fine procedura verifica_memorizza_esistenza
```

GENESIS OF THE PRIME NUMBERS

* Law of Genesis and Structure of the Prime Numbers

* The Algorithm “Method of the compass and the numerical range”

* Rudimentary implementation of the algorithm named “method of the compass and the numerical range” in language C/C++

ISMAELE ALONGI

ARRANGEMENT OF TRANSLATION FROM THE ORIGINAL ONE IN ITALIAN TO ENGLISH.
THE ORIGINAL VERSION IS IN ITALIAN LANGUAGE.

GENESIS OF THE PRIME NUMBERS

Law of Genesis and Structure of the Prime Numbers

"The beauty, the wisdom, the power and the glory belong to God, the Most High, He Who makes becoming.

He gives to us the privilege and the joy to be able to know His eternal intention for the humanity."

Ismaele Alongi.

From an analysis of the prime numbers I have deduced the following Law of Genesis and Structure of the Prime Numbers:

In the numerical succession of the prime numbers, every new prime number is born in the first numerical passage left empty by the numerical progressions of the precedents known prime numbers.

Originally I had expressed this concept this way:

Every new prime number is produced in the first space left empty by the inferior numerical series.

The numerical succession of the prime numbers lies on a half-line that has origin in the field of the ended one and point to the endless one. Ideally I have imagined some numerical successions superimposed that they have origin from situated points on a straight line imaginary orthogonal to the half-line of the natural numbers and passing for the point zero. It is extraordinary to imagine all the prime numbers lined up on this imaginary straight line: virtually we imagine an immense and endless composed number that goes to fill the endless one... or it could be also the endless prime number... However is, there are interrogative that for the man they are inscrutable, really because the mind human work in the field of the ended one and, since the endless one cannot be filled with this method, to try to close the circle endless take us to error.

The idea of the new empty spaces that they are filled as from the new prime numbers, it springs from the observation of the special even prime number: the number two.

If in fact we observe the behavior of the progression of the number two, we immediately notice that it goes to fill ½ of endless of the natural numbers. What would it be necessary to fill the rest of the endless one? A numerical progression in base one with reason two. It is not this however the logic of the succession of the prime numbers. The endless one is gradually filled, and since the endless one cannot be filled, it will never be filled, therefore for the human mind endless prime numbers exist. It will always stay in fact a first passage numerical void over the progressions superimposed of the known prime numbers: that first passage numerical void is one “zero”, at all banal, as loved to say Bernhard Riemann, and that numerical void is occupied by a new prime number by to know, a new numerical planet with one precise cyclical trajectory of its, a frequency with a precise length of wave.

It doesn't want there so much to risk idea that exploring the prime numbers along the half-line of the natural numbers, we are exploring only half the endless one, the other half it is found to the mirror, on the virtual half-line of the negative numbers, therefore very probably we witness a mathematical mirage if we think about seeing on the virtual alignment that passes for the zero-origin an only and endless composed number, instead probably in the mirage there is an endless prime number.

I don't know if these my hypotheses go to show the Riemann's hypothesis, that almost seems to me a succeeded mathematical hazard, but this result that I have gotten it is born from another search that I am still developing: the factorization of the composed numbers.

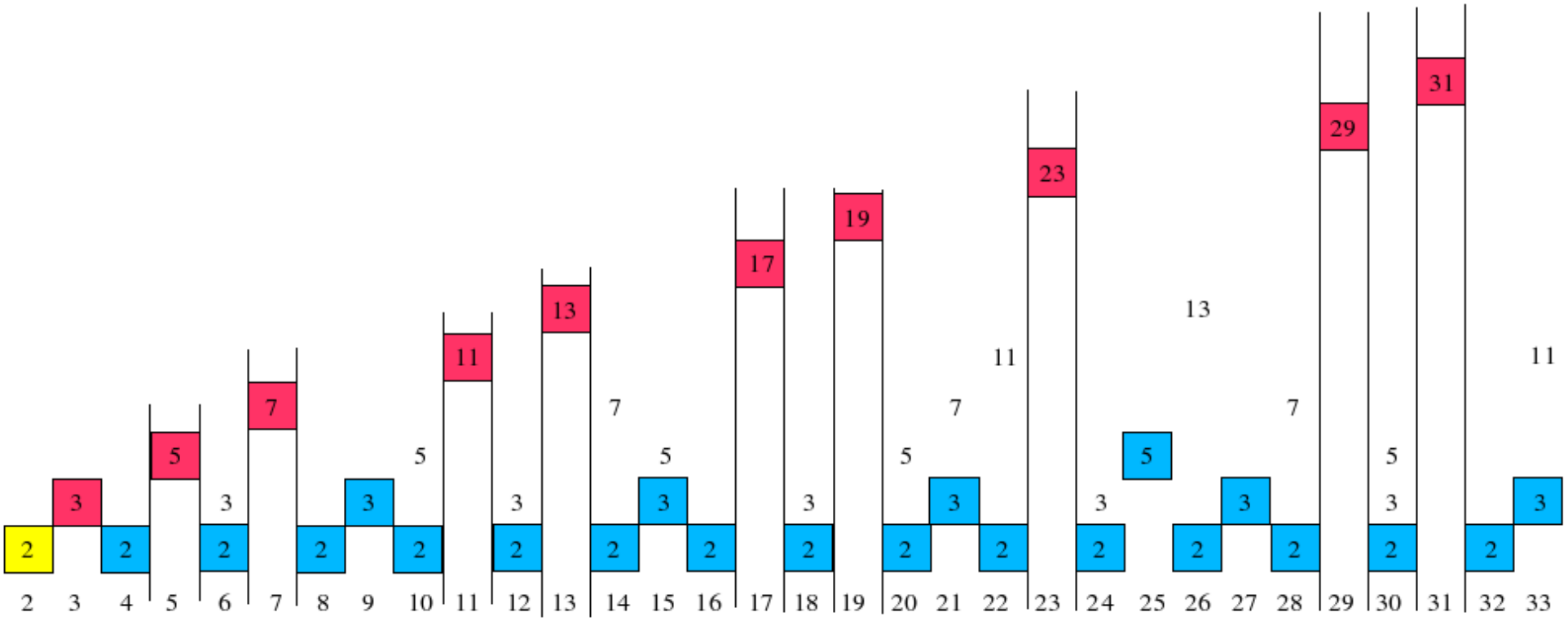
I took the law of numerical succession and genesis of the prime numbers, that I have deduced, as starting point for developing an enough effective system to calculate the following prime numbers in a datum numerical interval, gives the precedents known prime numbers. This algorithm I have denominated it method of the numerical compass and the numerical range.

As element of base would be enough the fundamental number two, to get all the other prime numbers, in fact if we observe the progression of the number two, the first space numerical void is between two and four, therefore the next prime number is three. We observe then the progressions superimposed of two and of three: the next numerical passage is five etc.

To visualize better the concept I have realized a simple graph. My discovery in fact it was born in the moment in which I have begun to consider for the varied natural numbers that I have analyzed in succession, the least divisor - great of the unity - relatively to every number, therefore I have begun to glimpse some symmetries that they were cyclically repeated, almost as a fantastic numerical genetic sequence: the DNA of the prime numbers.

Law of Genesis and Structure of the Prime Numbers

Graphic representation



GENESIS OF THE PRIME NUMBERS

The Algorithm “Method of the compass and the numerical range”

Initial Datum
Sequence of known prime numbers.
We know the sequence of the prime numbers from 2 to “n.”

Phase of calculation 0
Center of symmetry and ray of symmetry
The progressions produced by the varied prime numbers from 2 to “n” they create “symmetries”, therefore we decide to consider the position of the last known prime number “n” as “center of the symmetry”, or as center of the half-circumference traced that have as center the point “n”. The value of the last known prime number we will also consider it as “ray of symmetry.”

Elements of the progression of the number 2 inside the ray of symmetry.
How many elements will it have the progression with base 2 and reason 2 inside the ray of symmetry? The number of elements (following named “n_two”) it will be equal to: (“ray_of_symmetry”–1) / 2.
How much the value of the last element of the 2 progression will amount (following named “last_two”) inside the ray of symmetry?
“last_two” = 2 + (“n_two”–1) * 2.
This calculation on the formula is founded for the arithmetic progressions:

$$a_n = a_1 + (n - 1) * d;$$

where a_n is the last element of the progression, a_1 is the first element of the progression, n is the number of elements of the progression, d is the reason.

Phase of calculation 1
Modules of the incomplete trajectories
For every known prime number to calculate the module (or rest) of the division among “center of symmetry” and the examined prime number, therefore to memorize the module that represents, for the relative prime number to it associated, its incomplete trajectory inside the ray of symmetry.

Phase of calculation 2
Lacking trajectories
For every prime number to calculate difference between the examined prime number and the relative module: this difference is the lacking trajectory of the relative prime number examined inside the “numerical range” over the last known prime number, however before 2n.

Phase of calculation 3

Creation of the progressions of the odd prime numbers those are inferior to "n" inside the "numerical range" in base to the lacking trajectories

For every prime number to calculate the elements of its own progression inside the numerical range (that range is equal to the ray of symmetry) in base to the lacking trajectories and in base to the following rules:

- if the lacking trajectory is even, the base of the progression is equal to the lacking trajectory;
- if the lacking trajectory is odd, the base of the progression is equal to the lacking trajectory + the relative prime number;
- to calculate the progression up to that the last element of the progression is inferior or equal to the "two progression last element" that we have calculated in precedence ("last_two"), according to the following rule of calculation that allows us to jump the composed even numbers those are inside the numerical range: $k_y = k_x + 2 * p$; where k_y represents the following element of the examined numerical progression, k_x the precedent element of the progression in matter, p represents the prime number related to the progression that we are examining;
- we memorize the elements of the varied progressions calculated inside the "numerical range": these are the intersections with the half-line of the natural numbers.

Phase of calculation 4

Calculation of the zeros, positions of the new prime numbers

We calculate every 2 numerical progression element inside the numerical range up to the limit of symmetry, in base zero and reason 2, therefore in progression we verify the existence of intersections with the half-line of the natural numbers: if the considered element exists in the preceding numerical progressions, it wants to say that the intersection already exists with the half-line of the natural numbers, therefore we are in presence of a composed number; if instead the two progression element - that we are examining - it doesn't exist in the preceding progressions, it wants to say that we are in presence of a numerical void, an empty space or of one "zero" not banal, as Riemann would say. The sum of the position of the zero + the prime number "n" (or the last known prime number that we have also named "center of symmetry") it gives as resulted the following prime number in the succession of the prime numbers. To continue with the calculations up to the limit of symmetry, because we will very probably find further prime numbers inside the numerical range.

Notes

In the case in which we had to not have found new prime numbers inside the numerical range (which is equal to the ray of symmetry), to proceed with a new numerical range. In base to the Bertrand's postulate, however, it is reasonable to hypothesize rather that inside the numerical range equal to the ray of symmetry (or up to the position 2n) we will always find some prime numbers.

I have performed a rudimentary implementation of this algorithm of calculation in language C/C++ and it is possible to perform the program to verify partly the functionalities of this method of calculation, or better clarifying ideas on as it must be formulated really the calculations.

Example of application of the denominated algorithm “method of the compass and the numerical range.”

Datum: It's given the succession of the prime numbers from 2 to 19.

Phase of calculation 0
Center of symmetry and ray of symmetry.

```
n = 19;     // n = element[8];
center_of_symmetry = 19;   // center_of_symmetry = n;
```

Elements of the progression of the number 2 inside the ray of symmetry.

```
n_two = (19–1) / 2;              // n_two = (n–1) / 2;
```

Phase of calculation 1
Modules of the incomplete trajectories

```
// module[i] = n% element[i];

module[2] = 19% 3;      // module[2] = 1;
module[3] = 19% 5;      // module[3] = 4;
module[4] = 19% 7;      // module[4] = 5;
module[5] = 19% 11;     // module[5] = 8;
module[6] = 19% 13;     // module[6] = 6;
module[7] = 19% 17;     // module[7] = 2;
```

Phase of calculation 2
Lacking trajectories

```
// LT[i] = element[i] - module[i];

LT[2] = 3   - module[2]; // LT[2] = 3 – 1;    -->  2
LT[3] = 5   - module[3]; // LT[3] = 5 – 4;    -->  1
LT[4] = 7   - module[4]; // LT[4] = 7 – 5;    -->  2
LT[5] = 11 - module[5]; // LT[5] = 11 – 8;  -->  3
LT[6] = 13 - module[6]; // LT[6] = 13 – 6;  -->  7
LT[7] = 17 - module[7]; // LT[7] = 17 – 2;  --> 15
```

Phase of calculation 3
Creation of the progressions of the odd prime numbers those are inferior to “n” inside the “numerical range” in base to the lacking trajectories

progression of the number 3, in base to the lacking trajectory:
base = 2; // LT[i] it is even, therefore: base=LT[i];
reason = 2 * element[i];

then we get 2, 8, 14, ~~20~~;

progression of the number 5, in base to the lacking trajectory:
base = 1 + element [i]; // LT[i] it is odd, therefore: base=LT[i]+element[i];
reason = 2 * element[i];

then we get 6, 16, ~~26~~;

progression of the number 7, in base to the lacking trajectory:
2, 16, ~~30~~;

progression of the number 11, in base to the lacking trajectory:
14, ~~36~~;

progression of the number 13, in base to the lacking trajectory:
~~20~~;

progression of the number 17, in base to the lacking trajectory:
~~32~~;

Phase of calculation 4
Calculation of the zeros, positions of the new prime numbers

In base to the phase of calculation 3, we have noticed that the followings intersections exist with the half-line of the natural numbers inside the numerical range:
2,8,14;
6,16;
2,16;
14.

The progression in base zero and reason two up to the limit of symmetry n, produces 9 elements:
2, 4, 6, 8, 10, 12, 14, 16, 18.

The followings numbers don't exist among the intersections, they are the first empty spaces or passages numerical denominated zeros or the positions of the new prime numbers inside the numerical range: 4, 10, 12, 18.

The formula therefore it is simple: zero = n + position;

// element[i] = n + position;

element[9] = n + 4; // 23 = 19 + 4;
element[10] = n + 10; // 29 = 19 + 10;
element[11] = n + 12; // 31 = 19 + 12;
element[12] = n + 18; // 37 = 19 + 18;

ATTENTION!!! READ WITH CARE THE FOLLOWING CONDITIONS OF USE BEFORE CONTINUING IN THE READING OF THE DOCUMENT.

ARRANGEMENT OF TRANSLATION FROM THE ORIGINAL ONE IN ITALIAN TO ENGLISH.
THE ORIGINAL VERSION IS IN ITALIAN LANGUAGE.

Accord of License of use to non exclusive title between the owner of the intellectual ownership and the final consumer.

(It make reference to the original Italian license)

The intellectual owner is Ismaele Alongi (following suitable as “the author”), been born in 1977 in Italy, author of the work object of the intellectual ownership that is titled “Genesis of the prime numbers - Rudimentary implementation of the algorithm named “method of the compass and the numerical range” in language C/C++”.

The aforesaid work object of the intellectual ownership is constituted by the program for computer and the algorithm in it expressed and represented, both are been conceived, created and written from the author Ismaele Alongi (following suitable as “the following program for computer and the algorithm in it represented”), which work is found in the following pages of this document.

The final consumer is whoever decides to read, to use or to employ the following program for computer and the algorithm in it represented, accepting without reservations the conditions of this accord of license of use.

The following program for computer and the algorithm in it represented are granted in license of use to non exclusive title “so for as they are” and without any implicit or explicit guarantee, for one only consumer and for one only computer. The following program for computer and the algorithm in it represented they are not freeware, on the contrary they are simply shown to the public for better expressing and to make better comprehensible the ideas of the author with the purpose to communicate those ideas to the public.

Whoever doesn't agree with the conditions of this accord of license, is not authorized to read and/or to use in some way the following program for computer and the algorithm in it represented.

Who agrees instead with the conditions of the present accord of license, it arranges with the author that the final consumer won't ask some type of reimbursement for direct or indirect damages for the use proper and/or improper than contained and/or performed by the following program for computer and from the algorithm in it represented and for what is in it expressed and/or contained, neither the final consumer has to bring damages to other subjects through the use or application of the following program for computer and of the algorithm in it represented. The final consumer that agrees with the present accord of license, which consumer decides to read and/or to perform and/or to use and/or to employ in public the following program for computer and the algorithm in it represented, has to communicate to the people that directly or indirectly they will enjoy some application that derives of it, the conditions of the present accord of license and to put such people under the conditions to individually decide and to express his/her/their own decision, with the purpose to accept or to refuse the conditions of the present accord of license, therefore it will achieve the authorized use of it or less, in base to the express conditions in the present accord of license.

The author reserve to himself all methods and all way of economic use of his own intellectual ownership (in special way for the algorithm) and the final consumer that agrees with the present accord of license expressly consents to this limitation of use.

The reading and/or the use and/or the employment, in any way and with any method, of the followings program for computer and of the algorithm in it represented, those actions behave from the final consumer explicit acceptance and without reservations of the express conditions in the present accord of license of use to non exclusive title between the owner of the intellectual ownership and the final consumer.

GENESIS OF THE PRIME NUMBERS

Rudimentary implementation of the algorithm named “method of the compass and the numerical range” in language C/C++

RESERVED INTELLECTUAL AND LITERARY OWNERSHIP


```
// "The beauty, the wisdom, the power and the glory belong to God,
// the Most High, He Who makes becoming. He gives to us the joy and the
// privilege to be able to know His eternal intention for the humanity."
//                                                    Ismaele Alongi.

// One first rudimentary implementation of a verification of the "Law of Genesis and
// Structure of the prime numbers", hypothesized by Ismaele Alongi, and of the algorithm
// of the “compass and of the numerical range” of the author Ismaele Alongi.
// Other names to be proposed are: "Law of numerical succession of the prime numbers"
// or "order of numerical progression of the prime numbers"

//-------------------------------------------------------------------------
// Notes on the operation of the program: the program uses as input a
// file of text with the initial prime numbers, therefore it produces as output
// a file of text with the prime numbers individualized in the following order
// to those initial.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include <limits.h>
#include <values.h>
#include <float.h>
#include <iostream>
#include <fstream>

#define FALSE 0
#define TRUE 1

using namespace std;

void initialization_lists(int elements[]);
void verify_memorizing_existence(int ex, int existence[]);
void numeric_compass(int elements[]);

main ()
{
int initial_primes[1000]; //  array of the initial prime numbers

initialization_lists(initial_primes);// initialization of the list of the primes

numeric_compass(initial_primes);
                            // function of calculation of the new prime numbers:
                             // rudimentary implementation, see details.
} // end main program
```

```
//---------------------------------------------------------------------------
void numeric_compass(int elements[])
{
// declaration of variables
FILE *destination;

int modules[1000], // array of the modules
              LT[1000], // array of the lacking trajectories
              existence[1000]; // array of the  intersections

int center_of_symmetry, i, j,
              base, k, n,
              n_two, last_two,
              position = 0, zero;

short intersection=FALSE;

existence[0]=0;

destination=fopen("new_primes.txt", "w");

//  phase of calculation 0: center of symmetry
n=elements[0]; //  nth prime known
center_of_symmetry=elements[n];//  it is the last known prime number

cout << "Performed calculation 0: center of symmetry\n";

// elements of the progression of the number two inside the ray of symmetry
n_two=(center_of_symmetry - 1) / 2; //  number of elements of the
                                    //  two progression

last_two= 2 + (n_two - 1) * 2; // formula: An = A1 + (n - 1) * d;
                               // where An is the nth term looked for,
                               // A1 is the first element of the progression,
                               // n is the number of elements of the progr.,
                               // d is the reason.
cout << "Performed calculation elements of the progression of the number two"
     << "inside the ray of symmetry.\n";

for (i=2; i<=n; i++)
    cout << "Known prime numbers, "<< i << " = " << elements[i] <<"\n";
    cout << "other "<< elements[20]<< "other " << elements[0]<<"\n";

for (i=2; i<=n; i++)
    {
      // phase of calculation 1:  modules of the incomplete trajectories
      modules[i]=center_of_symmetry%elements[i];//esegue il module e memorizza
      cout << " Performed calculation 1: modules of the incomplete trajectories "
           << " \n modules[" << i << "] = " << modules[i] << "\n";

      // phase of calculation 2:  lacking trajectories
      LT[i]=elements[i]-modules[i];
                      // it calculates the lacking trajectories and it memorizes

      cout << " Performed calculation 2: lacking trajectories \n"
           << "LT[" << i << "] = " << LT[i] << "\n";
```

```
      // phase of calculation 3: creation of the numerical progressions;
      // progressions of the remainders prime numbers,
      // in base to the lacking trajectories

      if (LT[i]%2==0)
         {base=LT[i];}
         // even case: base of progression = lacking trajectory

         else base = LT[i] + elements[i];
         // odd case:
         // base progr. = lacking trajectory + relative prime number

      k=base;

      cout << "base = " << base << "\n";

      if (k == last_two)
          verify_memorizing_existence(k, existence);

      if (k < last_two)
         { verify_memorizing_existence(k, existence);

           while (k < last_two)
                 { k = k + 2 * elements[i]; // we jump the coincidences with
                                            // the composed even numbers
                   if (k<=last_two)
                       verify_memorizing_existence(k, existence);
                 }; // end "while"
         }; // end "if (k< last_two)"

      cout << " Performed calculation 3: creation of the numerical progressions"
           << "\n progressions of the remainders prime numbers,in base to the"
           << "lacking trajectories\n";

      cout << "index of the for " << i << "\n";

    }; // end "for" of the phases of calculation 1, 2, 3

// phase 4: calculation of the zeros, positions of the new prime numbers
// - we calculate the numerical progression of the 2 up to the limit of symmetry
// - we verify the existence of the elements of the numerical progression:
//   --if the number doesn't exist, this is the "zero", therefore we perform
//     the sum: center_of_symmetry + position;
// - the cycle finishes when it is reached the limit of symmetry
// Noteworthy: it is necessary to keep on refining the program for when the new
// prime numbers will be rarer.... because I don't know if inside the
// range of the symmetry they are always there new prime numbers, even if
// the hypothesis could be risked that is always.

for (i=1; i<=n_two; i++)
    {
     position=position + 2; // we are calculating the elements of the
                            // two progression, one to the time

     for (j=1; j<=existence[0]; j++)
         if (position==existence[j])
             { intersection=TRUE;
               j=existence[0];
             };

     if (intersection==FALSE)  // once closed the nested "for" it calculates
                               // the zero if there is the condition of void
        { zero=center_of_symmetry + position;
          fprintf (destination,"%ld\n", zero);
        }
       else intersection=FALSE;
```

```
      cout << "Performed calculation 4: calculation of the zeros, positions of"
           << "the new prime numbers\n...index of the for "<< i <<"\n";

     }; // end of the “for” of the two progression “as to fill”
        // inside the ray of symmetry

// Noteworthy: If new prime numbers are not found inside the ray
// of symmetry, it is necessary to continue the search with a new “numerical
// range", up to that we don't find them. This option is to implement.

fclose(destination);
} // end function “numeric_compass”

//---------------------------------------------------------------------------
void initialization_lists(int elements[])
{
FILE *origin;

int x=0;

origin=fopen("known_primes.txt", "r");

while (fscanf(origin, "%ld", &elements[x+1]) !=EOF)
      x++;

fclose(origin);

elements[0]=x; // index of array

} // end procedure “initialization_lists”

//---------------------------------------------------------------------------
void verify_memorizing_existence(int ex, int existence[])
{
short duplicated=FALSE;
int w;

if (existence[0]==0)
   { existence[1]=ex;
     existence[0]=1;
   }
   else
        { for (w=1; w<=existence[0]; w++)
             if (existence[w]==ex)
                { duplicated=TRUE;
                  w=existence[0];
                }; // end “for--if”

             if (duplicated==FALSE)
                { existence[0]++;
                  w=existence[0];
                  existence[w]=ex;
                };
        }; // end “else”
} // end procedure verify_memorizing_existence
```