```
/*
 * Corso di Fondamenti di Informatica
 * Esercizio:
 * ripasso di quanto visto a lezione: confronto tra gli algoritmi di
 * ricerca sequenziale e binaria
 */

/* Il programma popola una lista sequenziale con i primi N numeri
primi, in ordine crescente di valore. Successivamente, il programma verifica
se i numeri inseriti via tastiera da un utente sono primi, ricercandoli
nella lista dapprima con l'uso dell'algoritmo di ricerca sequenziale e poi
utilizzando l'algoritmo di ricerca binaria. (La ricerca viene fatta due volte,
con metodi diversi, per evidenziare le differenze tra i due metodi a fini
didattici.)

L'algoritmo di ricerca binaria è in grado di trovare l'elemento cercato (o di
stabilire che esso non è presente nella lista) eseguendo un numero di confronti
(ovvero di verifiche di uguaglianza tra l'elemento cercato e uno degli
elementi del campo array della lista) pari -nel caso pessimo- a

                  log (N)
                     2

Al confronto, l'algoritmo più banale (ricerca lineare o sequenziale: ovvero
confrontare il valore cercato con tutti quelli contenuti nella lista, uno dopo
l'altro) richiede -sempre nel caso pessimo- N confronti.
Dunque, in particolare per N grande, la ricerca binaria risulta molto più
efficiente, dal momento che, al crescere di N, log2(N) diventa rapidamente
molto minore di N.
Si noti che la ricerca binaria e' utilizzabile solo per la ricerca di elementi
in elenchi ORDINATI.*/

/* NOTA: un numero naturale è un numero primo se è maggiore di 1 e possiede
 * due e solo due divisori interi: il numero stesso e il numero 1.
 * I numeri primi sono infiniti: in ordine crescente, essi includono i numeri
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59... */

/* NOTA: nel programma sono state lasciate, a fini didattici, le istruzioni
 * printf "di debug" utilizzate per verificare il suo corretto funzionamento e
 * per facilitare la correzione degli errori.
 * Sono state trasformate in commenti, rendendole di fatto ininfluenti sul
```

```
 * funzionamento del programma, ma non eliminate. */


#include <stdio.h>
#define DIM_ELENCO 2000
/* numero di elementi dell'elenco dei numeri primi generato dal programma:
 * quello identificato con N nell'introduzione */

int main()
{
  struct
  {
    int NumeriPrimi[DIM_ELENCO];
    /* contiene i primi DIM_ELENCO numeri primi, in ordine crescente */
    int NumElementi;
    /* numero di elementi utilizzati del campo array della lista sequenziale */
  } ListaPrimi;
  /* elenco dei numeri primi */

  int NumeroDaValutare;
  /* numero candidato ad essere inserito nell'elenco dei numeri primi */
  int DivisoreAttuale;
  /* valore attualmente in esame come possibile divisore del numero candidato */
  int IlNumeroEPrimo;
  /* solo se IlNumeroEPrimo = 1 il numero candidato e' effettivamente primo;
   * qualsiasi altro valore indica che il numero non e' primo */
  /* NOTA: una scelta ancora piu' chiara sarebbe stata definire il tipo di dato
   * boolean (la definizione di nuovi tipi di dato verrà spiegata più avanti
   * nel corso), e poi dichiarare IlNumeroEPrimo come di tipo boolean. Il tipo
   * boolean serve a definire variabili che possono assumere 2 soli valori:
   * false e true. */
  int DaCercare;
  /* numero inserito dall'utente, da ricercare nell'elenco dei numeri primi */
  int ElementoAttuale;
  /* elemento del campo array della lista sequenziale correntemente in esame
   * durante la ricerca sequenziale */
  int NumeroIterazioni;
  /* numero di confronti tra numeri eseguiti dall'algoritmo di ricerca: usato
   * per confrontare l'efficienza degli algoritmi di ricerca sequenziale e
   * binaria */
  int Min, Max;
```

```
/* minimo e massimo indice della porzione di array nella quale si ricerca il
 * numero fornito dall'utente durante la ricerca binaria */
int Med;
/* indice dell'elemento centrale dell'intervallo compreso tra Min e Max
 * durante la ricerca binaria */


/*****************************************************************/
/* Generazione elenco numeri primi. Per farlo riempiamo il campo array
   della lista sequenziale a partire dall'elemento di indice 0, con numeri
   che verifichiamo essere primi. Per verificare se un numero è primo
   applichiamo la definizione, verificando che nessuna delle divisioni
   intere tra tale numero e gli interi positivi ad esso inferiori abbia
   resto nullo. */

ListaPrimi.NumElementi = 0;
NumeroDaValutare = 2;
/* per essere primo un numero deve essere intero e maggiore di 1, per cui
 * cominciamo a valutare i possibili candidati partendo dal numero 2 */

while (ListaPrimi.NumElementi < DIM_ELENCO)
{
  /***** controlliamo se tra i numeri compresi tra 1 e NumeroDaValutare
   (estremi esclusi) esistono divisori interi di NumeroDaValutare: se ne
   esiste almeno uno il numero non e' primo (per la definizione di numero
   primo) *****/

  DivisoreAttuale = NumeroDaValutare - 1;
  IlNumeroEPrimo = 1;
  /* per ora assumiamo che il numero sia primo: eventualmente smentiremo
   * l'ipotesi più avanti */

  while ( (DivisoreAttuale > 1) && (1 == IlNumeroEPrimo) )
  /* la seconda parte della condizione serve ad evitare di eseguire ulteriori
   * divisioni quando si e' verificato che il numero in esame non e' primo */
  {
    if (0 == NumeroDaValutare%DivisoreAttuale)
                /* se NumeroDaValutare e' divisibile per DivisoreAttuale, ovverosia se
                 * il resto della divisione intera tra i due è zero */
                {
                  IlNumeroEPrimo = 0;
```

```
                    /* printf("\n###DEBUG %d e' divisibile per %d, dunque non e' primo.",
                    NumeroDaValutare, DivisoreAttuale); */
                  }
      --DivisoreAttuale;
    }

    /* Al termine dell'esecuzione di questo ciclo, IlNumeroEPrimo e' ancora
     * pari ad 1 solo se non sono stati trovati divisori interi diversi dal
     * numero in esame e da 1: cioe' se NumeroDaValutare e' primo.
     * Si noti che nel caso particolare in cui il NumeroDaValutare vale 2 il
     * ciclo non viene eseguito nemmeno una volta, dal momento che il valore
     * iniziale di DivisoreAttuale, ovvero NumeroDaValutare-1, è pari ad 1. */

    if (1 == IlNumeroEPrimo)
    {
      /* scriviamo il numero primo appena trovato nella lista sequenziale: */
      ListaPrimi.NumeriPrimi[ListaPrimi.NumElementi] = NumeroDaValutare;
      ++ListaPrimi.NumElementi;
    }

    ++NumeroDaValutare;
  }

  /* printf("\n###DEBUG Elenco dei primi %d numeri primi: ", DIM_ELENCO);
  for (Min=0; Min<DIM_ELENCO; ++Min) printf("%d ", ListaPrimi.NumeriPrimi[Min]); */

  printf("\nHo generato un elenco dei primi %d numeri primi.\nInserisci un numero intero compreso tra 2 e %d per sapere se
e' primo [per terminare inserisci un numero <0]:  ",
  DIM_ELENCO, ListaPrimi.NumeriPrimi[DIM_ELENCO-1]);

  scanf("%d", &DaCercare);

  while (DaCercare >= 0)
  /* il ciclo si interrompe se l'utente inserisce un numero negativo */
  {
    /*****************************************************************/
    /* Applichiamo dapprima la ricerca sequenziale al campo array della lista
     * sequenziale. Esamineremo tutti gli elementi dell'array (a partire dal
     * primo e proseguendo in ordine), confrontando ciascuno di essi con il
     * numero cercato, e fermandoci se
```

```
 * si verifica una qualsiasi delle seguenti condizioni:
 * 1. l'elemento dell'array attualmente in esame è più grande del numero
 *    cercato: dunque questo non può essere contenuto nell'array, perche'
 *    in caso contrario sarebbe stato gia' trovato;
 * 2. il contenuto dell'array e' stato esaminato per intero.
 * Si noti che alla fine della ricerca possono verificarsi due condizioni:
 * 1. l'elemento dell'array in corrispondenza del quale si è fermata la
 *    ricerca e' uguale al numero cercato, che dunque e' primo;
 * 2. l'elemento dell'array in corrispondenza del quale si è fermata la
 *    ricerca non e' uguale al numero cercato, che dunque non e' primo */

ElementoAttuale = 0;
NumeroIterazioni = 1;
/* si noti che NumeroIterazioni non è necessario per la ricerca, ed e'
 * stato introdotto solo per evidenziare la differenza tra ricerca
 * sequenziale e ricerca binaria */

while ( (ListaPrimi.NumeriPrimi[ElementoAttuale] < DaCercare)
        /* questa e' l'operazione di confronto tra numeri
         * considerata nel calcolo del numero di iterazioni */
        && (ElementoAttuale < DIM_ELENCO - 1) )
{
  ElementoAttuale = ElementoAttuale + 1;
  NumeroIterazioni = NumeroIterazioni + 1;
}

if (ListaPrimi.NumeriPrimi[ElementoAttuale] == DaCercare)
  {
    printf("Il numero %d e' primo.", DaCercare);
  }
  else
  {
    printf("Il numero %d non e' primo.", DaCercare);
  }

printf("\nL'algoritmo di ricerca sequenziale ha eseguito %d confronti.",
       NumeroIterazioni);

/*****************************************************************
 * RICERCA BINARIA
 * Ripetiamo ora l'operazione di ricerca applicando la ricerca binaria.
```

```
 * Ad ogni passo l'algoritmo prevede di cercare il numero DaCercare tra
 * gli elementi di ListaPrimi.NumeriPrimi aventi indice compreso tra Min e
 * Max (estremi inclusi). Col procedere della ricerca, Min e Max vengono via
 * via aggiornati, ripetendo il procedimento fino a che il primo e'
 * minore del secondo. Quando tale condizione non vale più, l'elemento
 * di indice Min viene confrontato con il numero cercato. Se e solo se i
 * due coincidono il numero cercato è nell'elenco. */

NumeroIterazioni = 0;

Min = 0;
Max = DIM_ELENCO-1;
/* inizialmente consideriamo l'intero elenco */

while (Min < Max)
{
  Med = Min + (Max - Min)/2;
  /* NOTE (si ricordi che quella soprastante e' una divisione tra interi):
   * 1. se Max-Min e' pari, Med e' l'elemento che si trova esattamente nel
   *    punto centrale dell'intervallo Min->Max (in questo caso,
   *    infatti, il numero di elementi del sottoelenco in esame, ovvero
   *    quello comprendente gli elementi di indici Min, Min+1, ..., Max e'
   *    dispari)
   * 2. se Max-Min è dispari tale elemento centrale non esiste (poiche' il
   *    sottoelenco comprende un numero pari di elementi) e Med e'
   *    l'elemento che si trova immediatamente prima del punto centrale
   *    dell'intervallo;
   * 3. nel caso particolare Max-Min=1, ovvero Max=Min+1, Med risulta
   *    coincidente con Min. */

  /* printf("\n###DEBUG array[Min=%d]=%d array[Med=%d]=%d array[Max=%d]=%d",
     Min, ListaPrimi.NumeriPrimi[Min], Med, ListaPrimi.NumeriPrimi[Med],
     Max, ListaPrimi.NumeriPrimi[Max]); */

  if (ListaPrimi.NumeriPrimi[Med] < DaCercare)
  /* questa e' l'operazione di confronto tra numeri
   * considerata nel calcolo del numero di iterazioni */
  {
    Min = Med + 1;
  }
  else
```

```
      /* in questo caso DaCercare <= ListaPrimi.NumeriPrimi[Med] */
      {
        Max = Med;
      }

      NumeroIterazioni = NumeroIterazioni + 1;
    }

    if (ListaPrimi.NumeriPrimi[Min] == DaCercare)
      {
        printf("\nIl numero %d e' primo.", DaCercare);
      }
      else
      {
        printf("\nIl numero %d non e' primo.", DaCercare);
      }

    printf("\nL'algoritmo di ricerca binaria ha eseguito %d confronti.",
           NumeroIterazioni);

    printf("\n----------------------");
    printf("\nInserisci un numero intero compreso tra 2 e %d per sapere se e' primo [per terminare inserisci un numero <0]:
", ListaPrimi.NumeriPrimi[DIM_ELENCO-1]);
    scanf("%d", &DaCercare);
  }

  return 0;
}



/*****************************************************************************
Traccia di un'esecuzione di prova del programma:

Ho generato un elenco dei primi 2000 numeri primi.
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  3
Il numero 3 e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 2 confronti
Il numero 3 e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
```

```
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  15
Il numero 15 non e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 7 confronti
Il numero 15 non e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  56
Il numero 56 non e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 17 confronti
Il numero 56 non e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  79
Il numero 79 e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 22 confronti
Il numero 79 e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  229
Il numero 229 e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 50 confronti
Il numero 229 e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  487
Il numero 487 e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 93 confronti
Il numero 487 e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  1246
Il numero 1246 non e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 204 confronti
Il numero 1246 non e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  14555
Il numero 14555 non e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 1706 confronti
Il numero 14555 non e' primo.
L'algoritmo di ricerca binaria ha eseguito 11 confronti
```

```
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  17388
Il numero 17388 non e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 2000 confronti
Il numero 17388 non e' primo.
L'algoritmo di ricerca binaria ha eseguito 10 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  17389
Il numero 17389 e' primo.
L'algoritmo di ricerca sequenziale ha eseguito 2000 confronti
Il numero 17389 e' primo.
L'algoritmo di ricerca binaria ha eseguito 10 confronti
----------------------
Inserisci un numero intero compreso tra 2 e 17389 per sapere se e' primo [per terminare inserisci un numero <0]:  -1
**************************************************************************/


/**************************************************************************
Esercizio: se l'utente inserisce un numero superiore al limite massimo
specificato dal programma, questo fornisce risposte corrette?
**************************************************************************/
```