```
// Psearch.cpp
// Creazione di thread in ambiente Windows
// Crea due thread per la ricerca di un valore fornito da tastiera
// all'interno di un vettore che viene inizializzato con valori random

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>

#define DIM 100000000
int v[DIM];       // globale

// struttura che definisce i parametri della funzione search
struct PAR
{
 int first_index;
 int last_index;
 int search_value;
};

// funzione che implementa il codice dei thread
unsigned long WINAPI search(void* arg)
{
  int t_Id; // identificatore del thread

  PAR *tmp=(PAR*)(arg); // tmp puntatore all'oggetto
  PAR par=*tmp;         // alla variabile par viene assegnata
                        // la struttura ricevuta come parametro della funzione
  t_Id = GetCurrentThreadId();  // Restituisce l'identificatore del thread

  unsigned long trovati=0;

  // Ricerca valore
  for (int i=par.first_index; i<=par.last_index; i++)
      {if(v[i]==par.search_value)
         {
           printf("T_ID %i -> trovato valore in posizione=%i \r\n",t_Id,i);
           trovati++;
         }
      }

  printf("\nT_ID %i ---------------------------> N.trovati %i \n",t_Id,trovati);

  ExitThread(0); // terminazione del thread
 }
```

```
// --------------  Programma principale
int main()
{
  int valore;
  struct PAR par_1, par_2;
  // si predispongono due handle per i thread che verranno creati
  HANDLE thread_1, thread_2;

  srand(time(NULL));
  //generazione casuale dei valori degli elementi del vettore
  for (int i=0; i<DIM; i++)
   {
     v[i]=rand()%1000000;
   }
  // richiesta valore da ricercare
  printf("Inserire il numero da ricercare (0-999999)");
  scanf(" %i", &valore);

  // Estremi della sezione di ricerca assegnati al primo thread
  par_1.first_index =0;
  par_1.last_index  = DIM/2-1;
  par_1.search_value=valore;

  // Estremi della sezione di ricerca assegnati al secondo thread
  par_2.first_index = DIM/2;
  par_2.last_index  = DIM-1;
  par_2.search_value=valore;

  // creazione dei thread e attesa della loro terminazione
  thread_1=CreateThread(NULL,4096,&search,&par_1,0,NULL);
  thread_2=CreateThread(NULL,4096,&search,&par_2,0,NULL);

  WaitForSingleObject(thread_1,INFINITE);
  WaitForSingleObject(thread_2,INFINITE);

system("pause");

}
```