Titolo unità didattica: Approccio divide et impera [13]

Titolo modulo : Function in C per la ricerca binaria e per algoritmi di raddoppiamento [04-C]

Sviluppo di function (algoritmi iterativi) per la ricerca binaria e il calcolo di somma e massimo basati sull'approccio divide et impera

Argomenti trattati:

- ✓ function in C per la ricerca binaria (versione iterativa)
- ✓ function in C per la somma (versione iterativa, raddoppiamento)
- ✓ function in C per il massimo (versione iterativa, raddoppiamento)

Prerequisiti richiesti: AP-07-09-C, AP-07-10-C, AP-13-03-T

```
int ric_bin (char chiave, char elenco[], int n) {
 int mediano, primo, ultimo;
 primo = 0 ;
 ultimo = n-1 ;
 while (primo <= ultimo) {
   mediano = (primo + ultimo)/2 ;
   if (chiave == elenco[mediano])
       { return mediano ; }
     else if (chiave < elenco[mediano])
             { ultimo = mediano - 1 ; }
              else
             { primo = mediano + 1 ; }
 }
 return -1 ;
}
```

```
int ric_bin(char chiave,char elenco[],int n)
{
 int mediano, primo = 0, ultimo = n-1;
 while(primo <= ultimo)
 {
  mediano = (primo + ultimo)/2;
  if(chiave == elenco[mediano])
     return mediano;
   else if(chiave < elenco[mediano])
              ultimo = mediano-1;
           else
              primo = mediano+1;
 }
 return -1;
}
```

```c
#include <stdio.h>
#include <sting.h>
int ric_bin(char, char [],int );
void main()
{
 char elenco[100], cdr;
 int indice;
 printf("inserire l'elenco ordinato\n");
 gets(elenco);
 printf("inserire il carattere da ricercare: ");
 scanf("%c",&cdr);
 indice = ric_bin(cdr,elenco,strlen(elenco));
 if(indice >= 0)
   printf("chiave trovata al posto %d\n",indice);
 else
   printf("chiave non trovata\n");
}
```

```
int ric_bin_S(char chiave[],char *elenco[],int n)
{
 int mediano, primo = 0, ultimo = n-1;
 while(primo <= ultimo)
 {
  mediano = (primo+ultimo)/2;
  if(strcmp(chiave,elenco[mediano]) == 0)
     return mediano;
   else if(strcmp(chiave,elenco[mediano]) < 0)
             ultimo = mediano-1;
          else
             primo = mediano+1;
 }
 return -1;
}
```

```c
#include <stdio.h>
#include <string.h>
int ric_bin_S(char [],char *[],int );
void main()
{
 char giorno[10];
 int n,indice;
 char *settimana[7] =
 {"domenica","giovedi","lunedi","martedi",
  "mercoledi","sabato","venerdi"};
 printf("inserire il giorno da ricercare: ");
 gets(giorno);

/* CONTINUA*/
```

```
 ...
 n = 7;
 indice = ric_bin_S(giorno,settimana,n);
 if(indice >= 0)
 {
   printf("chiave trovata\n");
   printf("al posto %d\n",indice);
 }
 else
   printf("chiave non trovata\n");
}
```

```
int somma_radd(int a[],int n)
{
 int k,i;
 for (k=log2_I(n);k>0;k--)
    for (i=1;i<=pow(2,(k-1));i++)
        a[i-1] = a[2*(i-1)]+a[(2*i)-1];
 return a[0];
}
```

```
int somma_raddoppiamento(int a[], int n) {
 int i, k, somma;
 for (k=log2(n); k> 0; k=k-1) {
    for (i=1; i <= 2^(k-1); i++) {

            a[i-1] = a[2*(i-1)] + a[2*i-1] ;
    }
  }
 somma = a[0] ;
 return somma ;

}
```

```c
#include <stdio.h>
#include <math.h>
int somma_radd(int [],int);
int log2_I(int);
void main()
{
 int mio_vet[]= {2,15,1,-1,22,0,-2,11};
 int somma;
 somma = somma_radd(mio_vet,8);
 printf("somma degli elementi:\n");
 printf("%d\n",somma);
}
int log2_I(int n) /* parte intera di log2(n) */
{
 const double log_2=0.693147180559945;/*log(2)*/
 return (int)floor(log(n)/log_2);
}
```

```
int max_radd(int a[],int n)
{
 int k,i;
 for (k=log2_I(n);k>0;k--)
    for (i=1;i<=pow(2,(k-1));i++)
      a[i-1]=max_I(a[2*(i-1)],a[(2*i)-1]);
 return a[0];
}
```

```
/* function per il max di due int*/
 int max_I(int a, int b)
{
 if (a > b)
     return a;
  else
     return b;
}
```

```
int max_raddoppiamento(int a[], int n) {
 int i, k, massimo;
 for (k=log2(n); k> 0; k=k-1) {
    for (i=1; i <= 2^(k-1); i++) {

                a[i-1] = max(a[2*(i-1)] + a[2*i-1]) ;
    }
  }
 massimo = a[0] ;
 return massimo ;
}
```

```c
#include <stdio.h>
#include <math.h>
int max_radd(int [],int);
int max_I(int,int);
int log2_I(int);
void main()
{
 int mio_vet[]= {2,15,1,-1,22,0,-2,11};
 int massimo;
 massimo = max_radd(mio_vet,8);
 printf("massimo degli elementi:\n");
 printf("%d\n",massimo);
}
```