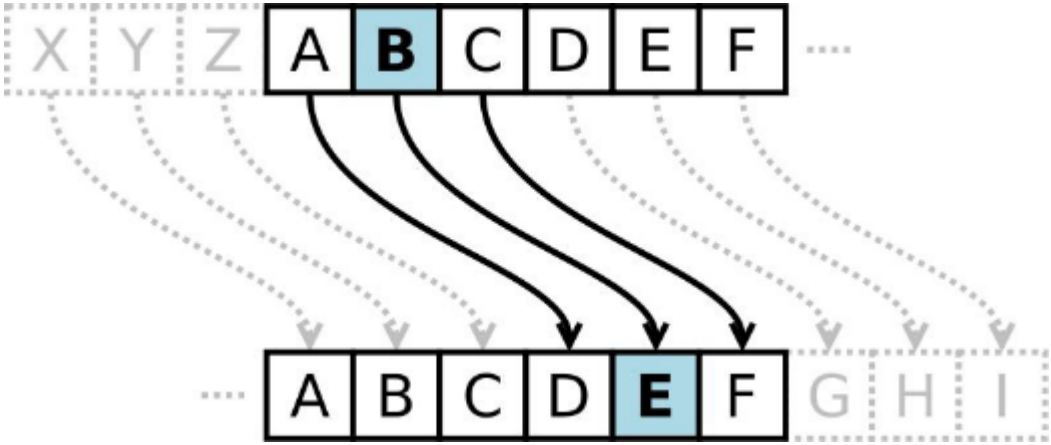
KDDE

Cifrario di Cesare, Ricerca all'interno di un array, Generazione numeri casuali

Dott. Emanuele Pio Barracchia

Cifrario di Cesare

- Il cifrario di Cesare è uno dei più antichi algoritmi crittografici
- Il cifrario di Cesare prende il nome da Giulio Cesare, che lo utilizzava per proteggere i suoi messaggi segreti.
- È un cifrario a sostituzione monoalfabetica in cui ogni lettera del testo in chiaro è sostituita nel testo cifrato dalla lettera che si trova un certo numero di posizioni *(la chiave)* dopo nell'alfabeto. In particolare, Cesare utilizzava 3 come chiave



Cifrario di Cesare

```
void encrypt(char message[], int key){
    char ch;
    for(int i = 0; message[i] != '\0'; ++i){
        ch = message[i];

        if(ch >= 'a' && ch <= 'z'){
            ch = ch + key;

            if(ch > 'z'){
                ch = ch - 'z' + 'a' - 1;
            }

            message[i] = ch;
        }
        else if(ch >= 'A' && ch <= 'Z'){
            ch = ch + key;

            if(ch > 'Z'){
                ch = ch - 'Z' + 'A' - 1;
            }

            message[i] = ch;
        }
    }
}
```

Cifrario di Cesare - Esercizio

- Scrivere un metodo in linguaggio C che, ricevendo in input il messaggio da codificare e la chiave, restituisca il messaggio cifrato.

- Cercare di decifrare il seguente messaggio: **fliudulr gl fhvduh** con chiave 3

Cifrario di Cesare - Esercizio

- Scrivere un metodo in linguaggio C che, ricevendo in input il messaggio da codificare e la chiave, restituisca il messaggio cifrato.

```
void decrypt(char message[], int key){
    char ch;
    for(int i = 0; message[i] != '\0'; ++i){
        ch = message[i];

        if(ch >= 'a' && ch <= 'z'){
            ch = ch - key;

            if(ch < 'a'){
                ch = ch + 'z' - 'a' + 1;
            }

            message[i] = ch;
        }
        else if(ch >= 'A' && ch <= 'Z'){
            ch = ch - key;

            if(ch < 'A'){
                ch = ch + 'Z' - 'A' + 1;
            }

            message[i] = ch;
        }
    }
}
```

Ricerca all'interno di un array

Esistono due algoritmi principali di ricerca di un elemento all'interno di un array:

- Algoritmo di ricerca sequenziale
- Algoritmo di ricerca dicotomica

Ricerca sequenziale

- L'algoritmo di ricerca sequenziale (o lineare) è un algoritmo utilizzabile per trovare un elemento in un insieme non ordinato
- Effettua una ricerca all'interno dell'array in modo sequenziale
- In particolare, controlla in sequenza gli elementi dell'array e verifica per ogni elemento controllato se è uguale all'elemento cercato
- L'algoritmo termina se:
  - L'elemento analizzato è l'elemento cercato
  - Ha controllato tutto l'array senza trovare l'elemento cercato

Ricerca sequenziale

```
void linear_search(int array[],int search){
    int i = 0;
    int dim = sizeof(array)/sizeof(array[0]);

    for (i = 0; i < dim; i++)
    {
        if (array[i] == search)
        {
            printf("%d trovato in posizione %d.\n", search, i+1);
            break;
        }
    }
    if (i == dim)
        printf("%d non è presente nell'array.\n", search);
}
```

Ricerca sequenziale - Versione ricorsiva

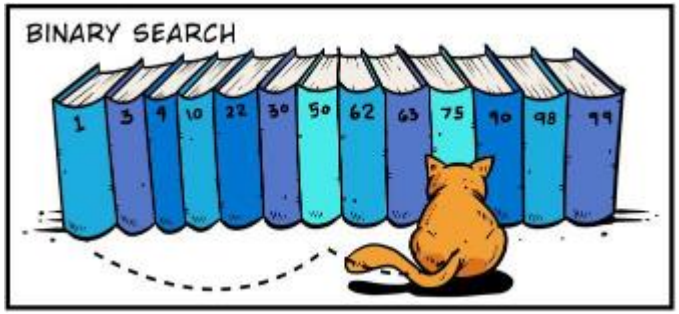
```c
#include<stdio.h>


int recSearch(int arr[], int left, int right, int x)
{
    if (right < left)
        return -1;
    if (arr[left] == x)
        return left;
    return recSearch(arr, left+1, right, x);
}

int main()
{
    int  search, n;

    printf("Inserire il numero di elementi dell'array: ");
    scanf("%d",&n);
    int array[n];

    printf("Inserire %d numeri: \n", n);

    for (int i = 0; i < n; i++)
        scanf("%d", &array[i]);

    printf("Inserire il numero da cercare: ");
    scanf("%d", &search);


    int index = recSearch(array, 0, n-1, search);

    if (index != -1)
        printf("L'elemento %d è presente in posizione %d. \n", search, index);
    else
        printf("L'elemento %d non è presente.", search);
    return 0;
}
```

Ricerca dicotomica

- L'algoritmo di ricerca dicotomica (o ricerca binaria) è un algoritmo di ricerca che individua l'indice di un determinato valore presente all'interno di un array ordinato

- Effettua mediamente meno confronti rispetto ad una ricerca sequenziale

- Il metodo alla base della ricerca binaria è lo stesso che si utilizza quando si cerca una parola all'interno del dizionario. L'idea è quella di iniziare la ricerca dal primo elemento, ma da quello centrale:
  - Se l'elemento corrisponde a quello cercato, la ricerca termina
  - Se è superiore, la ricerca viene ripetuta sugli elementi precedenti
  - Se è inferiore, la ricerca viene ripetuta su quelli successivi



Ricerca dicotomica

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

Ricerca dicotomica

First = 0

Middle = 2

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 2

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 2

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 2

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 2

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 0

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 0

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 0

Middle = 0

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 0

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 1

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 1

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 1

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 1

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

Ricerca dicotomica

First = 1

Middle = 1

Search = 5

```
void binary_search(int array[], int search){
    int dim = sizeof(array)/sizeof(array[0]);
    int first = 0;
    int last = dim - 1;
    int middle = (first+last)/2;

    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;
        else if (array[middle] == search) {
            printf("%d trovato in posizione %d.\n", search, middle+1);
            break;
        }
        else
            last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("%d non è presente nell'array.\n", search);

}
```

1	5	8	10	12

5 trovato in posizione 2.

Ricerca dicotomica - Versione ricorsiva

```
void binary_search_rec(int array[], int low, int high, int search)
{
    int mid;

    if (low > high)
    {
        printf("%d non trovato.\n", search);
        return;
    }
    mid = (low + high) / 2;
    if (array[mid] == search)
    {
        printf("%d trovato in posizione %d.\n", search, mid);
    }
    else if (array[mid] > search)
    {
        binary_search_rec(array, low, mid - 1, search);
    }
    else if (array[mid] < search)
    {
        binary_search_rec(array, mid + 1, high, search);
    }
}
```

Esercizio - Test primalità

Scrivere un programma in linguaggio C che, dato in input un numero naturale positivo, controlli se il numero è un numero primo.

NB. un numero si dice primo se è divisibile solo per 1 e per sé stesso.

Esercizio - Test primalità

```c
int main_prime()
{
    int n, i, flag = 0;

    printf("Inserire un numero positivo: ");
    scanf("%d",&n);

    for(i=2; i<=n/2; ++i)
    {
        // condition for nonprime number
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }

    if (flag==0)
        printf("%d è un numero primo.\n",n);
    else
        printf("%d non è un numero primo.\n",n);

    return 0;
}
```

Generazione numeri casuali

(Simpliest ever random number generator)

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Generazione numeri casuali

```c
int main() {
    int c, n;

    srand((unsigned)time(NULL));

    printf("10 numeri interi presenti in [1,100]\n");

    for (c = 1; c <= 10; c++) {
        n = rand() % 100 + 1;
        printf("%d\n", n);
    }

    return 0;
}
```

Esercizio 1

- Scrivere un programma in linguaggio C che legga da tastiera una sequenza di numeri positivi e ad ogni numero letto ne stampi la somma progressiva.

- Il programma termina quando si introduce un numero minore o uguale a zero.

Esercizio 1 - Soluzione

```c
#include <stdio.h>

int main()
{
    int num;
    int somma= 0;

    printf ("Inserisci un numero: ");
    scanf("%d", &num);

    while(num>=0)
    {
        somma = somma + num;
        printf("La somma progressiva attuale è pari a: %d\n", somma);

        printf ("Inserisci un numero: ");
        scanf("%d", &num);
    }
    printf("La somma progressiva finale è pari a: %d\n", somma);
    return 0;
}
```

Esercizio 2

Si chiedano 10 numeri in input. Verificare se i numeri inseriti sono pari o dispari.

Stampare a video prima tutti i numeri dispari, poi tutti i pari.

Esercizio 2 - Soluzione

```c
#include<stdio.h>

int main() {
    int array[10];
    int i;

    printf("Inserisci 10 numeri: \n");

    for (i = 0; i < 10; i++) {
        scanf("%d", &array[i]);
    }

    for (i = 0; i < 10; i++) {
        if (array[i] % 2 != 0) {
            printf("Numero dispari: %d \n", array[i]);
        }
    }

    for (i = 0; i < 10; i++) {
        if (array[i] % 2 == 0) {
            printf("Numero pari: %d \n", array[i]);
        }
    }

    return 0;
}
```

Esercizio 3

- Generare un numero a caso nell'intervallo [1,100] e chiedere all'utente un numero fino a quando non e' uguale a quello generato casualmente.
- Dire ogni volta se il numero immesso è > o < di quello iniziale.

Esercizio 3 - Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int num;
    int random;

    srand((unsigned)time(NULL));
    random = rand()% 100+1;
    num = random -1; //evito di impostare num pari a random

    while (num != random){
        printf("Inserisci un numero: ");
        scanf("%d", &num);
        if(num < random){
            printf("Inserisci un numero maggiore. \n");
        } else if (num > random){
            printf("Inserisci un numero minore. \n");
        }
    }
    printf("Numero trovato!");

}
```

Esercizio 4

Scrivere un programma in linguaggio C che:

- Riceve in input un numero ***n*** dall'utente
- Costruisce un array di numeri interi vuoto di dimensione ***n***
- Popola l'array con numeri casuali nell'intervallo **[-200, 99]**
- Stampa l'array popolato
- Cerca se è presente il numero **42** (utilizzare l'algoritmo di **ricerca binaria**)

Esercizio 4 - Soluzione

```c
#include <stdlib.h>
#include <stdio.h>

void binary_search_es1(int [], int, int, int);
void bubble_sort_es1(int [], int);

int main()
{
    int search, size, i;


    printf("Enter size of a list: ");
    scanf("%d", &size);
    int array[size];
    printf("Generating random numbers\n");
    for(i = 0; i < size; i++)
    {
        array[i] = (rand() % 300) - 200;
        printf("%d  ", array[i]);
    }
    bubble_sort_es1(array, size);
    printf("\n\n");
    printf("Enter key to search\n");
    scanf("%d", &search);
    binary_search_es1(array, 0, size, search);

}
```

```c
void bubble_sort_es1(int list[], int size)
{
    int temp, i, j;
    for (i = 0; i < size; i++)
    {
        for (j = i; j < size; j++)
        {
            if (list[i] > list[j])
            {
                temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
        }
    }
}
```

```c
void binary_search_es1(int array[], int low, int high, int search)
{
    int mid;

    if (low > high)
    {
        printf("%d non trovato.\n", search);
        return;
    }
    mid = (low + high) / 2;
    if (array[mid] == search)
    {
        printf("%d trovato in posizione %d.\n", search, mid);
    }
    else if (array[mid] > search)
    {
        binary_search_es1(array, low, mid - 1, search);
    }
    else if (array[mid] < search)
    {
        binary_search_es1(array, mid + 1, high, search);
    }
}
```

Domande?