Insertion sort in C++

In questa lezione affronteremo l'**insertion sort in C++**, un algoritmo di ordinamento che utilizza un metodo molto simile a quello usato da un essere umano per ordinare un mazzo di carte.

L'algoritmo di ordinamento insertion sort è molto semplice da implementare e non utilizza un array di appoggio, è dunque un algoritmo **in place**. In questo modo si risparmia memoria.

L'algoritmo utilizza due indici, ad esempio **i** e **j,** dove **i** punta al secondo elemento, mentre **j** punta al primo.

Quindi, per un ordinamento crescente, si confrontano i due elementi e se l'elemento puntato dall'indice j è maggiore dell'elemento puntato dall'indice i si scambiano, altrimenti l'indice avanza.

Il procedimento si ripete finché non si trova la posizione dove inserire l'elemento. In questo modo gli elementi maggiori vengono spostati verso destra.

Se volessimo ordinare l'array in ordine decrescente, dovremmo agire in maniera opposta.

Insertion sort – Esempio funzionamento

Facciamo subito un esempio per capire la logica di funzionamento.

Partiamo dunque da un array con questi elementi:

6 2 4 9 1 8

Innanzitutto confrontiamo 6 con 2

- dato che 6 è maggiore di 2 si sposta il 6 e si procede all'inserimento del 2.

Primo passaggio: 2 6 4 9 1 8

Adesso si confronta 6 con 4

- dato che 6 è maggiore di 4 si dovrebbero scambiare ma prima dobbiamo confrontare il 4 anche con il 2. Dato che 2 non è maggiore di 4 si può scambiare la posizione del 6 con quella del 4.

Secondo passaggio: 2 4 6 9 1 8

Dopo viene confrontato 6 con 9,

- dato che 6 non è maggiore di 9 non si fa nulla, si incrementa solo l'indice.

Terzo passaggio: 2 4 6 9 1 8

Poi si confronta 9 con 1

- dato che 9 è maggiore di 1 si continua a confrontare 1 con gli altri elementi. Quindi, in questo caso essendo tutti maggiori, si sposteranno verso destra lasciando libera la prima posizione dove poter inserire l'elemento.

Quarto passaggio: 1 2 4 6 9 8

Infine si confronta il 9 con il 8,

- dato che è superiore si procede a confrontare il numero 8 con il 4, ma essendo superiore scambiamo il 9 con il numero 8.

Quinto passaggio: 1 2 4 6 8 9

Sviluppo insertion sort in C++

Per realizzare l'algoritmo chiediamo innanzitutto di inizializzare un array **a[]** con i valori inseriti da tastiera.

Dopo, per l'ordinamento, si utilizza un ciclo esterno con indice **i** che parte da **1** e una variabile temporanea **temp** nella quale memorizziamo il secondo valore (a[1]). L'indice **j** parte da **0**.

Quindi inizialmente **i** punta al secondo elemento mentre **j** punta al primo. Dopo utilizziamo il **while** che si ripete finché è vera questa condizione: **il valore puntato dall'indice j è maggiore di temp e j>=0.**

All'interno del while memorizziamo il valore puntato dall'indice j nella posizione successiva a j e decrementiamo j di 1.

Quando si finisce il ciclo più interno (while) allora nella posizione j+1 memorizziamo il valore di temp.

Continuiamo finché si verifica questa condizione: **i<n**.

Ecco il listato completo dell'insertion sort in C++.

```
#include<iostream>
using namespace std;
#define MAX 10
int main()
{	int n, a[MAX], i, j, temp;
	cout<<"Quanti elementi?: ";
	cin>>n;
	for(i=0;i<n;i++)
			{cout<<"Elemento numero "<<i+1<<": ";
			 cin>>a[i];
			}

	cout<<"Ordiniamo l'array \n";
	for(i=1;i<n;i++)
		{ temp=a[i];
		  j=i-1;
		while((a[j]>temp) && (j>=0))
			{ a[j+1]=a[j];
			  j--;
			 }
		a[j+1]=temp;
		}
		cout<<"Array dopo l'ordinamento: \n";
		for(i=0; i<n; i++)
			cout<<a[i]<<" ";
	return 0;
}
```

Algoritmo insesrtion sort in C++ spiegato passo passo:

Partiamo da questo array non ordinato:

6 2 4 9 1 8

Il ciclo esterno verrà eseguito 5 volte, perchè partiamo dalla seconda posizione.

Prima iterazione

for(i=1;i<n;i++) // Inizialmente l'indice i parte da 1, cioè **i=1**

temp=arr[i]; //**temp=2**

j=i-1; //**j=0**

while(a[j]>temp && j>=0)

/*6>2 ***vero*** *j>=0* ***vero.*** Dunque la condizione è vera*/

a[j+1]=a[j]; //**a[1]=6**

j−; //j=-1.

/*Dato che la condizione j>=0 non è verificata si esce dal ciclo while*/

a[j+1]=temp; //**a[0]=2**

Ci troveremo quindi in questa situazione:

2 6 4 9 1 8

Seconda iterazione

for(i=1;i<n;i++) //i viene incrementato di 1, quindi parte da 2, cioè **i=2**

temp=a[i]; //**temp=4**

j=i-1; //**j=1**

while(a[j]>temp && j>=0)

/* 6>4 ***vero*** *j>=0* ***vero.*** Dunque la condizione è vera*/

a[j+1]=a[j]; //cioè ad a[2] assegno il valore di a[1]. Quindi **a[2]=6**

j−; //j=0

/*Dato che a[j]>temp cioè 2>4 non è verificata, non si continua con il while*/

a[j+1]=temp; // cioè **a[1]=4**

Questa dunque la nuova situazione:

2 4 6 9 1 8

Terza iterazione

for(i=1;i<n;i++) //i viene incrementato di 1, quindi parte da 3, cioè **i=3**

temp=a[i]; //**temp=9**

j=i-1; //**j=2**

while(a[j]>temp && j>=0) // 6>9 ***falsa.*** Dunque la condizione è ***falsa.***

/*Non si entra dentro al ciclo while perciò **a[2] rimane a 6**, quindi la prossima istruzione sarà:*/

a[j+1]=temp; //cioè **a[3]=9**

Quarta iterazione

for(i=1;i<n;i++) //i viene incrementato di 1, quindi parte da 4, cioè **i=4**

temp=a[i]; //**temp=1**

j=i-1; //**j=3**

while(a[j]>temp && j>=0)

/* 9>1 ***vero***; j>=0 ***vero***. Quindi la condizione è **vera*/**

a[j+1]=a[j]; //**a[4]=9**

j−; //j=2

/*Dato che a[j]>temp cioè 9>1 è verificata e j>=0 anche, si continua con il while*/

a[j+1]=a[j]; //**a[3]=6**

j−; //j=1

/*Dato che a[j]>temp cioè 6>1 è verificata e j>=0 anche, si continua con il while*/

a[j+1]=a[j]; //**a[2]=4**

j−; //j=0

/*Dato che a[j]>temp cioè 4>1 è verificata e j>=0 anche, si continua con il while*/

a[j+1]=a[j]; //**a[1]=2**

j−; //j=-1

/*Dato che a[j]>temp cioè 2>1 è verificata e j>=0 non è verificata, si esce dal while*/

a[j+1]=temp; //cioè **a[0]=1**

Come si può notare gli elementi 2, 4, 6, 9 sono stati shiftati a destra.

Quindi avremo:

1 2 4 6 9 8

Quinta iterazione

for(i=1;i<n;i++) // l'indice i parte da 5, cioè **i=5**

temp=arr[i]; //temp=8

j=i-1; //j=4

while(a[j]>temp && j>=0)

/*9>8 ***vero e j>=0 vero.*** Dunque la condizione è vera*/

a[j+1]=a[j]; //a[5]=9

j–; //j=3.

/*Dato che la condizione **a[j]>temp** non è verificata si esce dal ciclo while*/

a[j+1]=temp; //**a[4]=8**

Adesso l'indice i diventa 6 che non è minore di 6 (i<n) e quindi si esce dal for.

Ecco l'algoritmo ordinato:

1 2 4 6 8 9

Questa è l'implementazione dell'algoritmo insertion sort in C++ che volevo presentare in questa lezione.