Esercitazione N°3

Luca Zepponi

23 dicembre 2022

1. DESCRIZIONE DEL PROGRAMMA

Lo scopo principale del programma è l'utilizzo di una struttura ad albero binario per memorizzare una lista di stati, ordinarli secondo un criterio prestabilito e stampare l'albero opportunamente ordinato. Al fine di inserire tutti i dati all'interno dei vari nodi, si chiede l'inserimento da tastiera delle seguenti informazioni:

- una nazione;
- la sua capitale;
- il numero dei suoi abitanti;
- la sua superficie espressa in chilometri quadrati;
- la latitudine della capitale;
- la longitudine della capitale.

Più precisamente, il programma chiederà all'inizio dell'esecuzione di scegliere subito il criterio di ordinamento desiderato fra due possibilità (latitudine o longitudine), non sarà possibile modificare tale scelta in futuro, dopodiché sarà richiesta un'ulteriore scelta all'utente: inserire un nuovo stato, elencare gli stati in base all'ordinamento chiesto all'inizio del programma o terminare l'esecuzione. Tale scelta verrà richiesta al termine dell'inserimento di un nuovo stato o dopo aver stampato le nazioni fino a quando l'utente non deciderà di interrompere il programma.

Il criterio con cui si costruisce l'albero è il seguente:

- si parte dalla radice e si guarda la longitudine o la latitudine contenuta nel nodo;

- nel nodo di sinistra vanno memorizzate le capitali che si trovano o più a nord o più ad est;
- se il nodo è presente, si guarda il suo contenuto e si prosegue come il punto precedente;
- se il nodo non è presente, lo si inserisce.

Le librerie utilizzate sono:

- `<algorithm>`: per poter definire correttamente la funzione `str2STR`;
- `<iostream>`: per gli input e gli output;
- `<string>`: per poter utilizzare l'omonima classe.

Sono stati utilizzati anche i seguenti comandi al fine di semplificare la notazione:

- `using namespace std;`;
- `using std::string;`;
- `using std::transform;`;
- `using std::toupper;`.

1.1. DESCRIZIONE DELLE SINGOLE FUNZIONI

Il `main()` contiene le istruzioni necessarie a unire le singole funzioni. La prima che viene richiamata è `latOLong()` per decidere l'ordinamento da seguire (latitudine o longitudine). Segue un ciclo `do-while` che continua a ripetersi fino a quando l'utente non decide di interrompere il programma. Fino ad allora, tramite la funzione `menu()`, l'utente può scegliere cosa fare (inserire un nuovo nodo, stampare i vari nodi inseriti o uscire dal programma). Lo `switch` invece si occupa di gestire la scelta che l'utente fa con la funzione `menu()`.

```
1  int main() {
2      setlocale(LC_ALL, "");
3
4      // Descrizione del programma
5      descrizione();
6
```

```
7   // Dichiarazione variabile
8   char scelta;
9
10  // La scelta dell'ordinamento va effettuata una e una sola volta all'inizio
11  int ordine = latOLong();
12
13  do {
14    // scelta dell'operazione da effettuare
15    scelta = menu();
16
17    switch (scelta) {
18      case '1':
19        // Crea nodo
20        creaAlbero(ordine);
21        break;
22      case '2':
23        // Elenca albero
24        elencaNazioni(radice);
25        break;
26      case '0':
27        // Termina esecuzione
28        cout << "Uscita del programma..." << endl;
29        cout << "Fine." << endl;
30        break;
31      default:
32        break;
33    }
34
35    cout << "\n";
36  } while (scelta != '0');
37 }
```

La prima funzione che si incontra è

`void descrizione()`.[1]

```
1  void descrizione() {
2    cout << "%-------------------------%" << endl;
3    cout << "Descrizione del programma." << endl;
4    cout << "Il seguente codice permette di memorizzare una lista di" << endl;
5    cout <<"stati inserendo:" << endl;
```

[1] Il seguente codice è stato scritto in modo da farlo entrare nel rettangolo grigio.

```
  cout << "- la nazione;" << endl;
  cout << "- la capitale;" << endl;
  cout << "- il numero di abitanti;" << endl;
  cout << "- la superficie in chilometri quadrati;" << endl;
  cout << "- la latitudine della capitale" << endl;
  cout << "  (positiva a nord, negativa a sud);" << endl;
  cout << "- la longitudine della capitale" << endl;
  cout << "  (positiva ad est, negativa a ovest)." << endl;
  cout << "\n";
  cout << "Il programma chiederà all'inizio dell'esecuzione di" << endl;
  cout << "scegliere subito il criterio di ordinamento desiderato fra" <<endl;
  cout << "due possibilità (latitudine o longitudine), non sarà" << endl;
  cout << "possibile modificare tale scelta in futuro, dopodiché sarà" << endl;
  cout << "richiesta un'ulteriore scelta all'utente:" << endl;
  cout << "1 - inserire un nuovo stato;" << endl;
  cout << "2 - elencare gli stati in base all'ordinamento chiesto" << endl;
  cout << "  all'inizio del programma" << endl;
  cout << "0 - terminare l'esecuzione." << endl;
  cout << "Tale scelta verrà richiesta al termine dell'inserimento di" << endl;
  cout << "un nuovo stato o dopo aver stampato le nazioni fino a" << endl;
  cout << "quando l'utente non deciderà di interrompere il" << endl;
  cout << "programma." << endl;
  cout << "%-------------------------%" << endl;
  cout << "\n";
}
```

Tale funzione si occupa solo di fornire all'utente alcune informazioni riguardanti il programma stesso, per esempio cosa fa, cosa l'utente dovrà inserire e il funzionamento del programma.

La seconda funzione è

`int latOLong()`.

La funzione non prende in input nessun dato e inizia con la dichiarazione delle variabili che verranno usate.

```
// Scelta ordinamento albero
int latOLong() {
  // Variabile per decidere l'ordinamento
  char ordine;
  // Variabile per confermarela scelta dell'ordinamento
  char sicuro;
```

Il ciclo `do-while` che segue è stato inserito per consentite all'utente di cambiare la decisione che prenderà riguardo all'ordinamento dell'albero che vuole seguire. Le variabili `sicuro` e `ordine` sono state dichiarate di tipo `char` (non `int`) per coprire quanti più errori possibili da parte dell'operatore umano. In particolare è stato scelto di usare questo tipo di dato anziché un intero per garantire il controllo dell'inserimento anche nel caso venisse digitata una lettera al posto di un numero o viceversa. Una volta confermata la scelta dell'ordinamento, si dichiara una nuova variabile di tipo intero che assumerà valore `1` o `0` a seconda della scelta dell'utente.

```
7   // Controllo decisione di ordinamento
8   do {
9     // Scelta ordinamento
10    cout << "Vuoi ordinare secondo:" << endl;
11    cout << "\"0\" - la longitudine;" << endl;
12    cout << "\"1\" - la latitudine." << endl;
13    cout << "ATTENZIONE! La scelta è permanente." << endl;
14    cout << "Scelgi attentamente..." << endl;
15    cin >> ordine;
16
17    // ordine è char per controllare eventuali errori con lettere
18    while ((ordine != '0') && (ordine != '1')) {
19      cout << "Attenzione, inserimento non corretto." << endl;
20      cout << "Prova ancora: ";
21      // Rinserire ordine
22      cin >> ordine;
23    }
24
25    // controllo decisione ordinamento
26    cout << "Sei sicuro della tua scelta?" << endl;
27    cout << "Non potrai tornare più indietro..." << endl;
28    cout << "Digita \"s\" per confermare." << endl;
29    cout << "Digita un qualsiasi altro tasto per ripensarci." << endl;
30    cin >> sicuro;
31  } while (sicuro != 's');
32
33
34  // da Char a Int
35  int ordineInt;
36  if (ordine == '1') ordineInt = 1;
37  else ordineInt = 0;
38
39  // Stampa scelta effettuata
```

```
40    if (ordineInt) cout << "Ordino secondo la latitudine." << endl;
41    else cout << "Ordino secondo la longitudine." << endl;
42
43    return ordineInt;
44  }
```

La funzione termina stampando una stringa che informa l'utente della scelta che ha effettuato e restituendo il valore immesso da tastiera.

La parte di codice riportata sotto si occupa di controllare che l'utente abbia inserito o `0` o `1` per decidere se ordinare l'albero secondo la latitudine o la longitudine. Nel caso l'inserimento non risultasse corretto, ovvero nel caso in cui l'utente non abbia inserito né `1` né `0`, viene stampato una stringa che informa dell'errore e richiede di inserire nuovamente la scelta.

```
1     // ordine è char per controllare eventuali errori con lettere
2     while ((ordine != '0') && (ordine != '1')) {
3       cout << "Attenzione, inserimento non corretto." << endl;
4       cout << "Prova ancora: ";
5       // Rinserire ordine
6       cin >> ordine;
7     }
```

La terza funzione che compare è

`char menu()`,

la quale si occupa di tre cose:

1. informare l'utente delle tre possibili scelte che ha a disposizione:

   1 - inserire una nazione;

   2 - elencare le nazioni con il criterio di ordinamento precedentemente scelto;

   0 - terminare il programma.

2. restituire un dato di tipo `char` che ha immagazzinato la scelta dell'utente;

3. controlla gli inserimenti da tastiera per assicurarsi che venga scritto uno dei tre `char` che il programma sa come interpretare e in caso tale controllo avesse esito negativo, viene richiesto all'utente di riscegliere ristampando anche il `MENU`.

```
char menu() {
  char scelta;

  do {
    cout << "MENU" << endl;
    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
    cout << "1 - Inserisci nuova nazione " << endl;
    cout << "2 - Elenca le nazioni" << endl;
    cout << "0 - Esci" << endl;
    cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
    cout << "Scelta: ";
    cin >> scelta;
    } while ((scelta != '1') && (scelta != '2') && (scelta != '0' ));
    // Se si digita un carattere diverso da questi tre, il menu viene
    // ristampato per chiedere un nuovo inserimento

    return scelta;
}
```

A seguire si ha la funzione

`void creaAlbero(int ordine).`

In tale funzione come prima cosa troviamo la dichiarazione delle variabili che si useranno, poi si chiede all'utente di inserire da tastiera una serie di informazioni che vengono memorizzate nelle precedenti variabili che serviranno a costruire un nuovo `stato`. Fra i vari input è presente un `if`: se come nazione si sceglie "Italia", viene richiamata la funzione `scherzetto` spiegata in seguito. La funzione `str2STR` si occupa di convertire i caratteri della stringa in maiuscolo, in questo modo il controllo viene effettuato indiscriminatamente da come viene scritto "Italia".

```
// Crea albero
void creaAlbero(int ordine) {
  // Dichiarazione variabili
  string nazione;
  string capitale;
  unsigned long int abitanti;
  float superficie;
  float latitudine;
  float longitudine;

```

```
11  // L'utente inserisce tutte le informazioni da tastiera
12  cout << "Inserisci nazione: ";
13  cin >> nazione;
14  cout << "Inserisci la capitale: ";
15  cin >> capitale;
16  if (str2STR(nazione) == "ITALIA") capitale = scherzo(capitale);
17  cout << "Inserisci il numero di abitanti: ";
18  cin >> abitanti;
19  cout << "Inserisci la superficie [km^2]: ";
20  cin >> superficie;
21  cout << "Inserisci latitudine: ";
22  cin >> latitudine;
23  cout << "Inserisci longitudine: ";
24  cin >> longitudine;
```

A questo punto si assegna ad `iteratore` il valore di `radice` e successivamente si controlla che `radice` sia vuota, `radice = NULL`. In caso positivo, si crea un nuovo nodo[2] e si assegnano tutti i valori utilizzando le informazioni fornite in input precedentemente.

```
25  iteratore = radice;
26
27  // Se radice = NULL
28  if (!radice) {
29    // radice = new stato(...) non funziona
30    radice = new stato;
31    radice->nazione = nazione;
32    radice->capitale = capitale;
33    radice->abitanti = abitanti;
34    radice->superficie = superficie;
35    radice->latitudine = latitudine;
36    radice->longitudine = longitudine;
37    radice->ramoSx = NULL;
38    radice->ramoDx = NULL;
39  }
```

Nel caso in cui `radice` non sia vuota, ovvero `radice != NULL`, si prosegue. Finché `iteratore` non è vuoto si resta nel ciclo `while`. `if (ordine)` è la condizione che serve ad ordinare l'albero o per la latitudine o per la longitudine (la variabile `ordine` è l'unica che viene passata in input alla funzione).

[2] L'assegnazione è stata svolta in questo modo perché la notazione sintetica non funzione con Visual Studio Code.

```
40    // Finché iteratore è non vuoto
41    while (iteratore) {
42      // Ordinamento per latitudine
43      if (ordine) {
```

Nel caso in cui si scelga di ordinare secondo la latitudine, `ordine = 1`, occorre determinare se il nuovo nodo va inserito a sinistra o a destra. Supponiamo che la latitudine inserita dall'utente sia strettamente maggiore di quella puntata dall'iteratore. Se `ramoSx` puntato dall'iteratore non è vuoto, `iteratore` viene riassegnato, altrimenti viene creato un nuovo nodo con la funzione `creaRamoSx`. Se invece la latitudine inserita dall'utente fosse minore o uguale di quella puntata da `iteratore`, si va a guardare `ramoDx`, se non è vuoto si riassegna `iteratore`, altrimenti si crea un nuovo nodo con la funzione `creaRamoDx`. Se, invece, si sceglie di ordinare secondo la longitudine, `ordine = 0`, si fa una cosa analoga ma considerando la longitudine invece della latitudine. In tutti i casi, dopo aver creato un nuovo nodo, si pone `iteratore = NULL` per continuare a restare all'interno del `while`.

```
44        if (latitudine > iteratore->latitudine) {
45          // Latitudine inserit maggiore => nuovo ramo a sinistra
46          // Se ramoSx puntato da iteratore è non vuoto, riassegna iteratore
47          if (iteratore->ramoSx) iteratore = iteratore->ramoSx;
48          // altrimenti crea nuovo ramo
49          else {
50            //iteratore ->ramoSx = new stato(); non funziona
51            creaRamoSx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
52
53            iteratore = NULL;
54          }
55        }
56        else {
57          // Logitudine inserit maggiore => nuovo ramo a sinistra
58          // Se ramoSx puntato da iteratore è non vuoto, riassegna iteratore
59          if (iteratore->ramoDx) iteratore = iteratore->ramoDx;
60          // altrimenti crea nuovo ramo
61          else {
62            creaRamoDx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
63
64            iteratore = NULL;
65          }
66        }
67      }
```

```
68      else {// ordinamento per longitudine
69            // longitudine positiva a est, negativa a ovest
70        if (longitudine > iteratore->longitudine) {
71          // longitudine inserita minore
72          if (iteratore->ramoSx) iteratore = iteratore->ramoSx;
73          else {
74            creaRamoSx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine)
75
76            iteratore = NULL;
77          }
78        }
79        else {// longitudine inserita maggiore
80          if (iteratore->ramoDx) iteratore = iteratore->ramoDx;
81          else {
82            creaRamoDx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine)
83
84            iteratore = NULL;
85          }
86        }
87      }
88    }
89  }
```

La funzione

`string scherzetto(string capitale)`

prende in input la stringa `capitale` e verifica che sia uguale a `ROMA`. Questa funzione viene eseguita solo se la nazione inserita è `Italia`. Per considerare ogni possibile scrittura di `ROMA` (come `roma`, `Roma`, `ROMA`, ecc.), prima di confrontare il contenuto della variabile `capitale` con `ROMA`, ogni carattere della stringa contenuta in `capitale` è stato convertito in maiuscolo grazie alla funzione `str2STR`. In questo modo, finché la variabile interna `confronto` sarà `0`, la funzione stampa un ironico messaggio e chiede all'utente di inserire la vera capitale dell'Italia. Una volta inserita una nuova capitale si procede ad un nuovo confronto.

```
1  // Scherzetto
2  string scherzo(string capitale) {
3    bool confronto;
4    // Confronto capitale inserita con "ROMA"
5    confronto = str2STR(capitale) == "ROMA";
6    // Per migliorare il confronto, i caratteri della stringa inserita dall'utente
```

```
  // verranno convertiti in maiuscoli

  while (!confronto) {
    // Confronto fra stringhe: fonte https://www.techiedelight.com/it/compare-two-strings-in-cpp/

    // Se <capitale> != roma\dots{}
    if (!confronto) {
      cout << "No, " << capitale << " non è caput mundi." << endl;
      cout << "Scherzetto ;)" << endl;
      cout << "Inserisci quella vera ... " << endl;
      cout << "Inserisci la capitale: ";
      cin >> capitale;
    }
    confronto = str2STR(capitale) == "ROMA";
  }
  return capitale;
}
```

Le due funzioni

```
void creaRamoSx( stato* iteratore, string nazione, string
                 capitale, unsigned long int abitanti,
                 float superficie, float latitudine,
                 float longitudine  )
```

e

```
void creaRamoDx( stato* iteratore, string nazione, string
                 capitale, unsigned long int abitanti,
                 float superficie, float latitudine,
                 float longitudine  )
```

servono per:

- inizializzare un nuovo stato rispettivamente nel ramo sinistro e destro dell'iteratore;
- successivamente vengono assegnate le proprietà allo stato `ramoSx` e `ramoDx`, che a loro volta sono proprietà di `iteratore`.

```
// Crea ramo sinistro
void creaRamoSx(stato* iteratore, string nazione, string capitale, unsigned long int abitanti, float sup
  iteratore->ramoSx = new stato;
  iteratore->ramoSx->nazione = nazione;
  iteratore->ramoSx->capitale = capitale;
  iteratore->ramoSx->abitanti = abitanti;
  iteratore->ramoSx->superficie = superficie;
```

```
    iteratore->ramoSx->latitudine = latitudine;
    iteratore->ramoSx->longitudine = longitudine;
    iteratore->ramoSx->ramoSx = NULL;
    iteratore->ramoSx->ramoDx = NULL;
}
```

```
// Crea ramo destro
void creaRamoDx(stato* iteratore, string nazione, string  capitale, unsigned long int abitanti, f
    iteratore->ramoDx = new stato;
    iteratore->ramoDx->nazione = nazione;
    iteratore->ramoDx->capitale = capitale;
    iteratore->ramoDx->abitanti = abitanti;
    iteratore->ramoDx->superficie = superficie;
    iteratore->ramoDx->latitudine = latitudine;
    iteratore->ramoDx->longitudine = longitudine;
    iteratore->ramoDx->ramoSx = NULL;
    iteratore->ramoDx->ramoDx = NULL;
}
```

La funzione

`void elencaNazioni(stato* radice)`

prende in input il puntatore a radice e se radice non è vuota, percorre tutti i rami `ramoSx` fino ad arrivare all'ultimo disponibile e procede poi a stamparne il contenuto. Fatto questo la funzione richiama se stessa e passa ad esplorare il primo ramo `ramoDx` e procede così fino alla completa esplorazione dell'albero.

L'ultima funzione definita è

`string str2STR(string stringa)`,

la quale prende in input una stringa (in questo programma prende in input `capitale`) e inizia a leggere tutti i caratteri in essa contenuti e li converte nei rispettivi caratteri maiuscoli.

```
// stringhe in maiuscolo
// da "stringa a "STRINGA"
string str2STR(string stringa) {
  transform(stringa.begin(),stringa.end(),stringa.begin (),[]( unsigned char c){
    return toupper(c);
  });

  return stringa;
}
```

2. CODICE SORGENTE

```
1  // Esercitazione 3 : Alberi
2  // Studente: Luca Zepponi
3  // Programma che stampa e chiede l'inserimento da tastiera di:
4  // - una nazione;
5  // - la sua capitale;
6  // - il numero dei suoi abitanti;
7  // - la sua superficie in chilometri quadrati;
8  // - la latitudine della capitale;
9  // - la longitudine della capitale.
10 //
11
12 // Librerie richieste
13 #include <algorithm>
14 #include <iostream>
15 #include <string>
16
17 using namespace std;
18 using std::string;
19 using std::transform;
20 using std::toupper;
21
22 #define DPRINT(VAR) cout << #VAR << " = " << (VAR) << endl;
23
24 // Struttura
25 struct stato {
26   string nazione;
27   string capitale;
28   unsigned long int abitanti;
29   float superficie;
30   float latitudine;
31   float longitudine;
32
33   stato* ramoSx;
34   stato* ramoDx;
35 };
36
37 stato* radice = NULL;
38 stato* iteratore = NULL;
39
40 // Funzioni di istanza
```

```
41 void descrizione();
42 int latOLong();
43 char menu();
44 void creaAlbero(int ordine);
45 string scherzo(string capitale);
46 void creaRamoSx(stato* iteratore, string nazione, string capitale, unsigned long int abitanti, f
47 void creaRamoDx(stato* iteratore, string nazione, string capitale, unsigned long int abitanti, f
48 void elencaNazioni(stato* radice);
49 // https://www.delftstack.com/it/howto/cpp/how-to-convert-string-to-uppercase-cpp/
50 string str2STR(string stringa);
51
52
53
54 int main() {
55     setlocale(LC_ALL, "");
56
57     // Descrizione del programma
58     descrizione();
59
60     // Dichiarazione variabile
61     char scelta;
62
63     // La scelta dell'ordinamento va effettuata una e una sola volta all'inizio
64     int ordine = latOLong();
65
66       do {
67           // scelta dell'operazione da effettuare
68           scelta = menu();
69
70           switch (scelta)
71           {
72           case '1':
73               // Crea nodo
74               creaAlbero(ordine);
75               break;
76           case '2':
77               // Elenca albero
78               elencaNazioni(radice);
79               break;
80           case '0':
81               // Termina esecuzione
82               cout << "Uscita del programma..." << endl;
83               cout << "Fine." << endl;
```

```
84              break;
85          default:
86              break;
87          }
88
89          cout << "\n";
90      } while (scelta != '0');
91  }
92
93
94
95  // Descrizione del programma
96  void descrizione() {
97    cout << "%-------------------------%" << endl;
98    cout << "Descrizione del programma." << endl;
99    cout << "Il seguente codice permette di memorizzare una lista di" << endl;
100   cout <<"stati inserendo:" << endl;
101   cout << "- la nazione;" << endl;
102   cout << "- la capitale;" << endl;
103   cout << "- il numero di abitanti;" << endl;
104   cout << "- la superficie in chilometri quadrati;" << endl;
105   cout << "- la latitudine della capitale" << endl;
106   cout << "  (positiva a nord, negativa a sud);" << endl;
107   cout << "- la longitudine della capitale" << endl;
108   cout << "  (positiva ad est, negativa a ovest)." << endl;
109   cout << "\n";
110   cout << "Il programma chiederà all'inizio dell'esecuzione di" << endl;
111   cout << "scegliere subito il criterio di ordinamento desiderato fra" <<endl;
112   cout << "due possibilità (latitudine o longitudine), non sarà" << endl;
113   cout << "possibile modificare tale scelta in futuro, dopodiché sarà" << endl;
114   cout << "richiesta un'ulteriore scelta all'utente:" << endl;
115   cout << "1 - inserire un nuovo stato;" << endl;
116   cout << "2 - elencare gli stati in base all'ordinamento chiesto" << endl;
117   cout << "  all'inizio del programma" << endl;
118   cout << "0 - terminare l'esecuzione." << endl;
119   cout << "Tale scelta verrà richiesta al termine dell'inserimento di" << endl;
120   cout << "un nuovo stato o dopo aver stampato le nazioni fino a" << endl;
121   cout << "quando l'utente non deciderà di interrompere il" << endl;
122   cout << "programma." << endl;
123   cout << "%-------------------------%" << endl;
124   cout << "\n";
125 }
126
```

```
127
128
129 // Scelta ordinamento albero
130 int latOLong() {
131   // Variabile per decidere l'ordinamento
132   char ordine;
133   // Variabile per confermarela scelta dell'ordinamento
134   char sicuro;
135
136   // Controllo decisione di ordinamento
137   do {
138     // Scelta ordinamento
139     cout << "Vuoi␣ordinare␣secondo:" << endl;
140     cout << "\"0\"␣-␣la␣longitudine;" << endl;
141     cout << "\"1\"␣-␣la␣latitudine." << endl;
142     cout << "ATTENZIONE!␣La␣scelta␣è permanente." << endl;
143     cout << "Scelgi␣attentamente..." << endl;
144     cin >> ordine;
145
146     // ordine è char per controllare eventuali errori con lettere
147     while ((ordine != '0') && (ordine != '1')) {
148       cout << "Attenzione,␣inserimento␣non␣corretto." << endl;
149       cout << "Prova␣ancora:␣";
150       // Rinserire ordine
151       cin >> ordine;
152     }
153
154     // controllo decisione ordinamento
155     cout << "Sei␣sicuro␣della␣tua␣scelta?" << endl;
156     cout << "Non␣potrai␣tornare␣più indietro..." << endl;
157     cout << "Digita␣\"s\"␣per␣confermare." << endl;
158     cout << "Digita␣un␣qualsiasi␣altro␣tasto␣per␣ripensarci." << endl;
159     cin >> sicuro;
160   } while (sicuro != 's');
161
162
163   // da Char a Int
164   int ordineInt;
165   if (ordine == '1') ordineInt = 1;
166   else ordineInt = 0;
167
168   // Stampa scelta effettuata
169   if (ordineInt) cout << "Ordino␣secondo␣la␣latitudine." << endl;
```

```
170   else cout << "Ordino secondo la longitudine." << endl;
171
172   return ordineInt;
173 }
174
175
176
177 // Scelta da fare:
178 // scelta = 1: si inserisce una nuova nazione con tutte le informazioni;
179 // scelta = 2: si stampa l'albero con tutti i nodi finora creati;
180 // scelta = 0: si termina l'esecuzione del programma.
181 // Solo con scelta = 0 il programma termina, dopo la stampa dell'albero sarà
182 // comunque possibile inserire nuovi nodi.
183 char menu() {
184   char scelta;
185
186   do {
187     cout << "MENU" << endl;
188     cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
189     cout << "1 - Inserisci nuova nazione " << endl;
190     cout << "2 - Elenca le nazioni" << endl;
191     cout << "0 - Esci" << endl;
192     cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
193     cout << "Scelta: ";
194     cin >> scelta;
195     } while ((scelta != '1') && (scelta != '2') && (scelta != '0'));
196     // Se si digita un carattere diverso da questi tre, il menu viene
197     // ristampato per chiedere un nuovo inserimento
198
199     return scelta;
200 }
201
202
203
204 // Crea albero
205 void creaAlbero(int ordine) {
206   // Dichiarazione variabili
207   string nazione;
208   string capitale;
209   unsigned long int abitanti;
210   float superficie;
211   float latitudine;
212   float longitudine;
```

```
213
214     // Dichiarazione variabili
215     string nazione;
216     string capitale;
217     unsigned long int abitanti;
218     float superficie;
219     float latitudine;
220     float longitudine;
221
222     // L'utente inserisce tutte le informazioni da tastiera
223     cout << "Inserisci la nazione: ";
224     cin >> nazione;
225     cout << "Inserisci la capitale: ";
226     cin >> capitale;
227     if (str2STR(nazione) == "ITALIA") capitale = scherzo(capitale);
228     cout << "Inserisci il numero di abitanti: ";
229     cin >> abitanti;
230     cout << "Inserisci la superficie [km^2]: ";
231     cin >> superficie;
232     cout << "Inserisci latitudine: ";
233     cin >> latitudine;
234     cout << "Inserisci longitudine: ";
235     cin >> longitudine;
236
237     iteratore = radice;
238
239     // Se radice = NULL
240     if (!radice) {
241       // radice = new stato(...) non funziona
242       radice = new stato;
243       radice->nazione = nazione;
244       radice->capitale = capitale;
245       radice->abitanti = abitanti;
246       radice->superficie = superficie;
247       radice->latitudine = latitudine;
248       radice->longitudine = longitudine;
249       radice->ramoSx = NULL;
250       radice->ramoDx = NULL;
251   }
252
253     // Finché iteratore è non vuoto
254     while (iteratore) {
255       // Ordinamento per latitudine
```

```
256     if (ordine) {
257       if (latitudine > iteratore->latitudine) {
258         // Latitudine inserita maggiore => nuovo ramo a sinistra
259         // Se ramoSx puntato da iteratore è non vuoto, riassegna iteratore
260         if (iteratore->ramoSx) iteratore = iteratore->ramoSx;
261         // altrimenti crea nuovo ramo
262         else {
263           //iteratore ->ramoSx = new stato(); non funziona
264           creaRamoSx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
265
266           iteratore = NULL;
267         }
268       }
269       else {
270         // Latitudine inserita minore => nuovo ramo a destra
271         if (iteratore->ramoDx) iteratore = iteratore->ramoDx;
272         // altrimenti crea nuovo ramo
273         else {
274           creaRamoDx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
275
276           iteratore = NULL;
277         }
278       }
279     }
280     else {// ordinamento per longitudine
281       // longitudine positiva a est, negativa a ovest
282       // Longitudine inserita maggiore => nuovo ramo a sinistra
283       if (longitudine > iteratore->longitudine) {
284         // longitudine inserita minore
285         if (iteratore->ramoSx) iteratore = iteratore->ramoSx;
286         else {
287           creaRamoSx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
288
289           iteratore = NULL;
290         }
291       }
292       else {// longitudine inserita minore => nuovo ramo a destra
293         if (iteratore->ramoDx) iteratore = iteratore->ramoDx;
294       else {
295         creaRamoDx(iteratore, nazione, capitale, abitanti, superficie, latitudine, longitudine);
296
297         iteratore = NULL;
298       }
```

```
299         }
300       }
301     }
302 }
303
304
305
306 // Scherzetto
307 string scherzo(string capitale) {
308   bool confronto;
309   // Confronto capitale inserita con "ROMA"
310   confronto = str2STR(capitale) == "ROMA";
311   // Per migliorare il confronto, i caratteri della stringa inserita dall'utente
312   // verranno convertiti in maiuscoli
313
314   while (!confronto) {
315     // Confronto fra stringhe: fonte https://www.techiedelight.com/it/compare-two-strings-in-
316
317     // Se <capitale> != roma\dots{}
318     if (!confronto) {
319       cout << "No, " << capitale << " non è caput mundi." << endl;
320       cout << "Scherzetto ;)" << endl;
321       cout << "Inserisci quella vera ... " << endl;
322       cout << "Inserisci la capitale: ";
323       cin >> capitale;
324     }
325     confronto = str2STR(capitale) == "ROMA";
326   }
327   return capitale;
328 }
329
330
331
332 // Crea ramo sinistro
333 void creaRamoSx(stato* iteratore, string nazione, string capitale, unsigned long int abitanti, f
334   iteratore->ramoSx = new stato;
335   iteratore->ramoSx->nazione = nazione;
336   iteratore->ramoSx->capitale = capitale;
337   iteratore->ramoSx->abitanti = abitanti;
338   iteratore->ramoSx->superficie = superficie;
339   iteratore->ramoSx->latitudine = latitudine;
340   iteratore->ramoSx->longitudine = longitudine;
341   iteratore->ramoSx->ramoSx = NULL;
```

```
342     iteratore->ramoSx->ramoDx = NULL;
343 }
344
345
346
347 // Crea ramo destro
348 void creaRamoDx(stato* iteratore, string nazione, string  capitale, unsigned long int abitanti, float sup
349     iteratore->ramoDx = new stato;
350     iteratore->ramoDx->nazione = nazione;
351     iteratore->ramoDx->capitale = capitale;
352     iteratore->ramoDx->abitanti = abitanti;
353     iteratore->ramoDx->superficie = superficie;
354     iteratore->ramoDx->latitudine = latitudine;
355     iteratore->ramoDx->longitudine = longitudine;
356     iteratore->ramoDx->ramoSx = NULL;
357     iteratore->ramoDx->ramoDx = NULL;
358 }
359
360 // Funzione che stampa l'albero
361 void elencaNazioni(stato* radice) {
362     if (radice) {
363         // Stampa tutte le informazioni
364         elencaNazioni(radice->ramoSx);
365         cout << "Nazione " << radice->nazione << ";" << endl;
366         cout << "- " << "capitale " << radice->capitale << ";" << endl;
367         cout << "- " << "abitanti " << radice->abitanti << ";" << endl;
368         cout << "- " << "superficie " << radice->superficie << ";" << endl;
369         cout << "- " << "latitudine " << radice->latitudine << ";" << endl;
370         cout << "- " << "longitudine " << radice->longitudine << "." << endl;
371         elencaNazioni(radice->ramoDx);
372     }
373 }
374
375
376
377 // stringhe in maiuscolo
378 // da "stringa a "STRINGA"
379 string str2STR(string stringa) {
380     transform(stringa.begin(), stringa.end(), stringa.begin (),[]( unsigned char c){
381         return toupper(c);
382     });
383
384     return stringa;
```

```
385 }
```

3. RISULTATO

Di seguito sono riportate le foto del programma in esecuzione. Rispettivamente:

- le immagini 1 a fronte, 2 a pagina 24 e 3 a pagina 25 riportano il funzionamento con ordinamento tramite latitudine;
- le immagini 4 a pagina 26, 5 a pagina 27 e 6 a pagina 28 riportano il funzionamento con ordinamento tramite longitudine.

4. OSSERVAZIONI E CONCLUSIONI

Una prima modifica che si potrebbe fare riguarda la variabile `ordineInt` presente nella funzione `int latOLong()`. Tale variabile potrebbe essere omessa, andrebbe però ridefinita la funzione e fare in modo che restituisca un `char`, dopodiché anche la funzione `creaAlbero` dovrebbe prendere in input, non un intero, ma un `char` e per finire andrebbe modificato opportunamente anche l'`if` che si occupa dell'ordinamento dell'albero. Non sono modifiche complesse da fare, ma per questioni di tempo mi limito solo a citarle qui.

Un'altra accortezza che si potrebbe prendere riguardo gli inserimenti da tastiera sono gli eventuali spazi che sono presenti nelle nazioni o nelle capitali. Non avendo a disposizione un computer (perché rotto), mi sono dovuto affidare ad un editor online[3]. Tale editor non mi ha permesso di utilizzare il comando `getline` per migliorare questo aspetto, infatti nelle prove riportate ho evitato di inserire anche il Regno Unito.

Un altro aspetto che può essere sicuramente migliorato è la creazione di nuovi rami. Prima che il computer si rompesse, usavo Visual Studio Code per programmare, ma tale compilatore non accettava la versione sintetica di `new` e per evitare di riscrivere le stesse righe di codice ho deciso di creare le funzioni `creaRamoSx` e `creaRamoDx`. Ho anche provato a creare una sola funzione ma non ci sono riuscito.

[3]L'editor utilizzato è quello del seguente link: `https://www.onlinegdb.com/online_c++_compiler`.

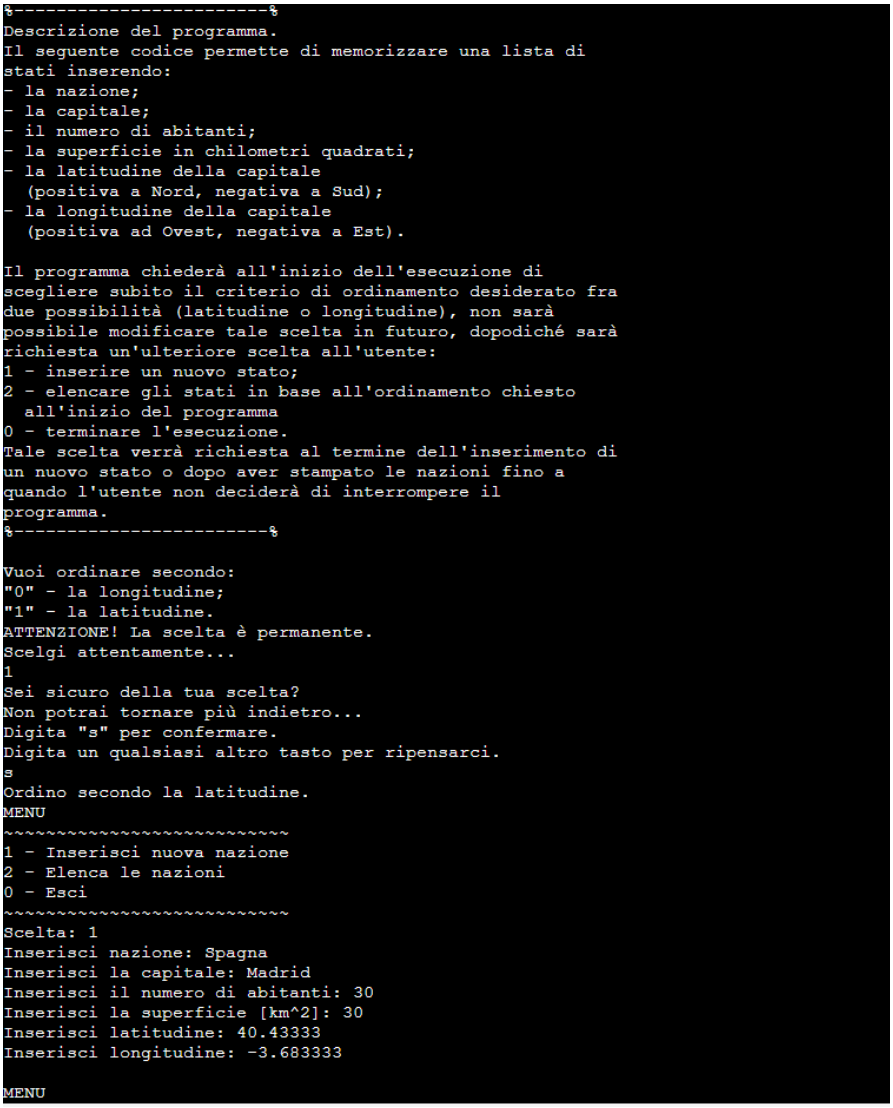

Figura 1: Ordinamento per latitudine, 1.

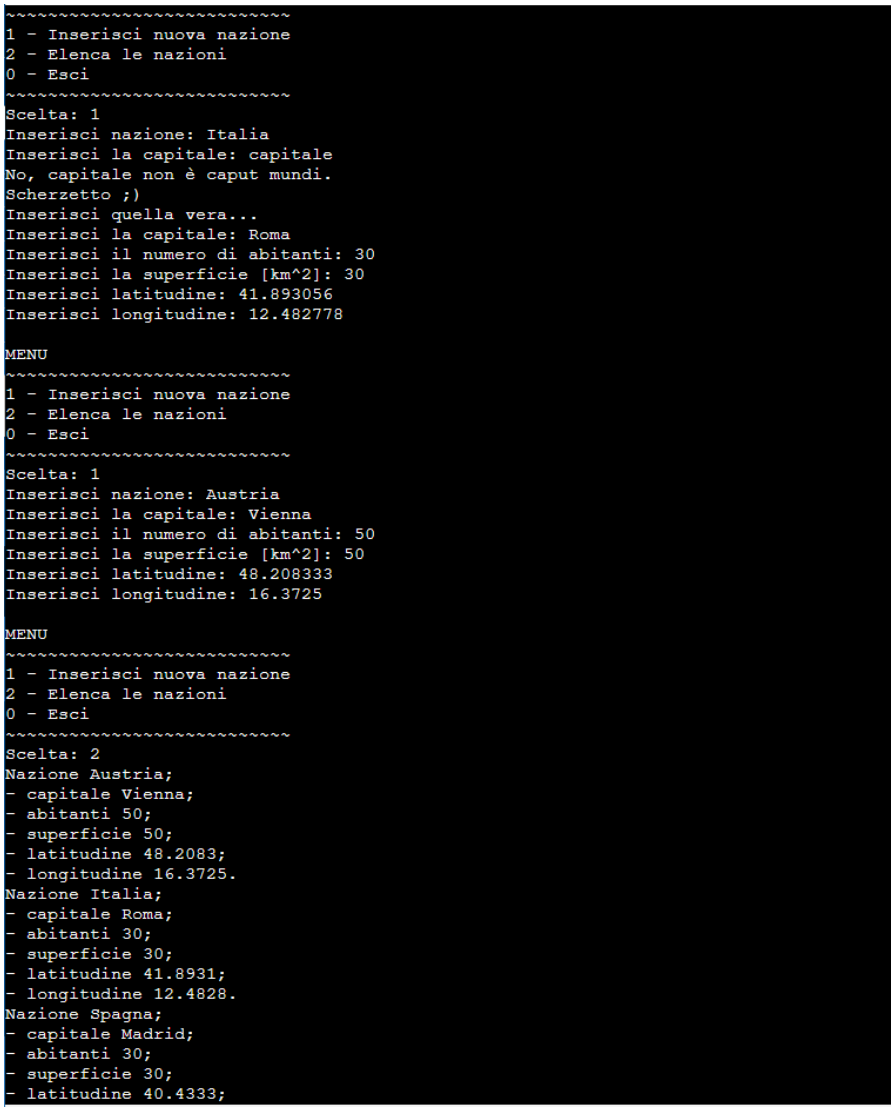

Figura 2: Ordinamento per latitudine, 2.

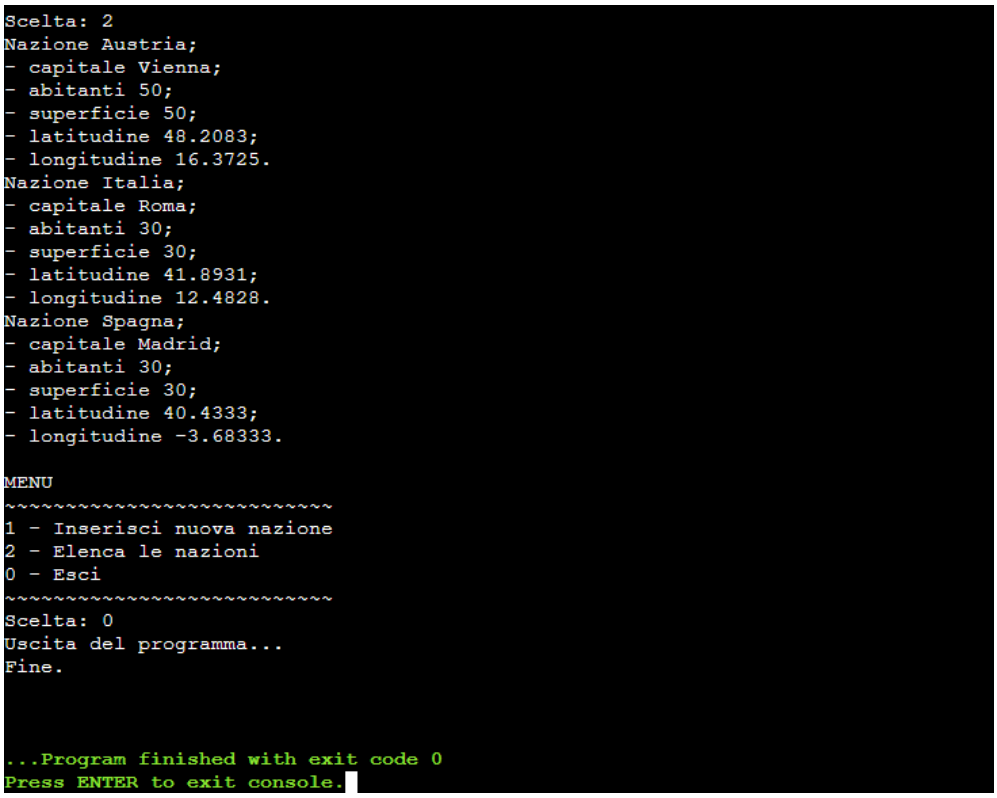

Figura 3: Ordinamento per latitudine, 3.

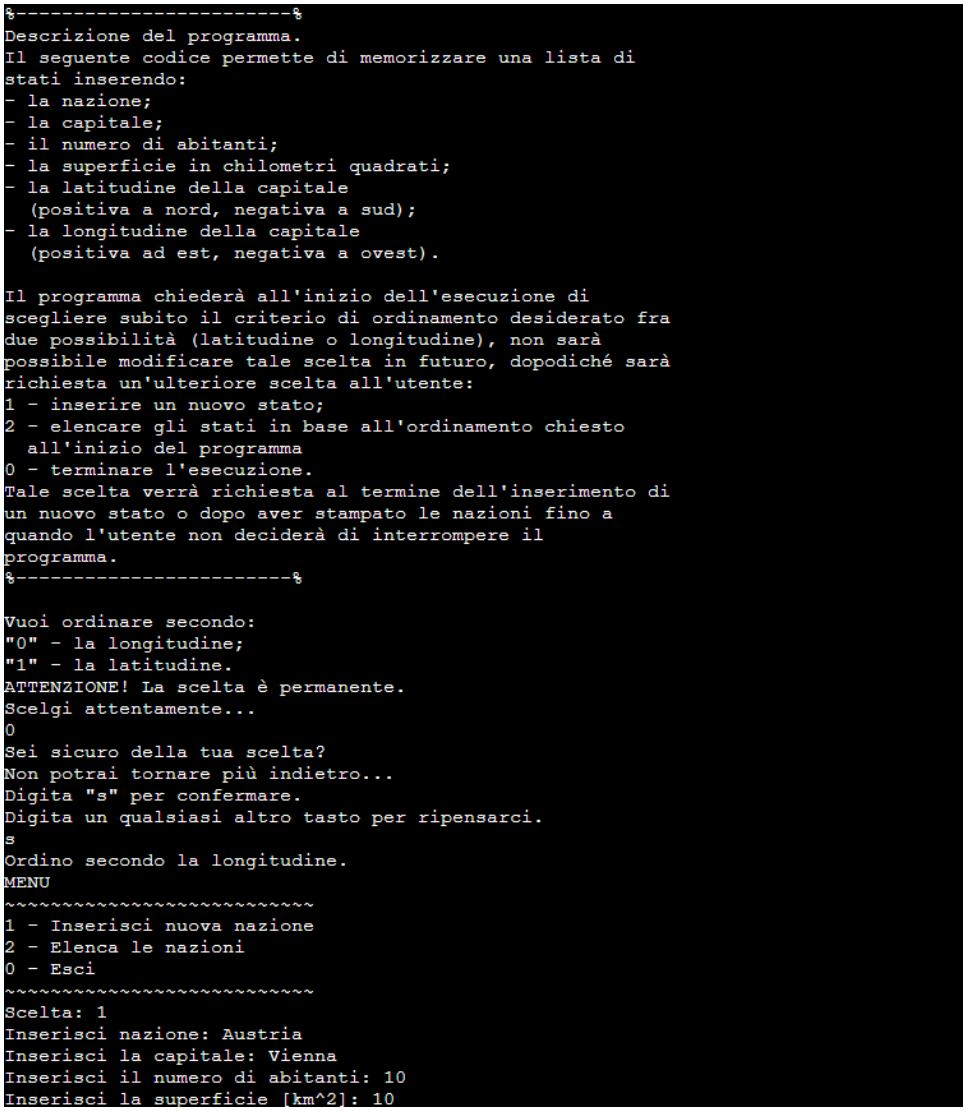

Figura 4: Ordinamento per longitudine, 1.

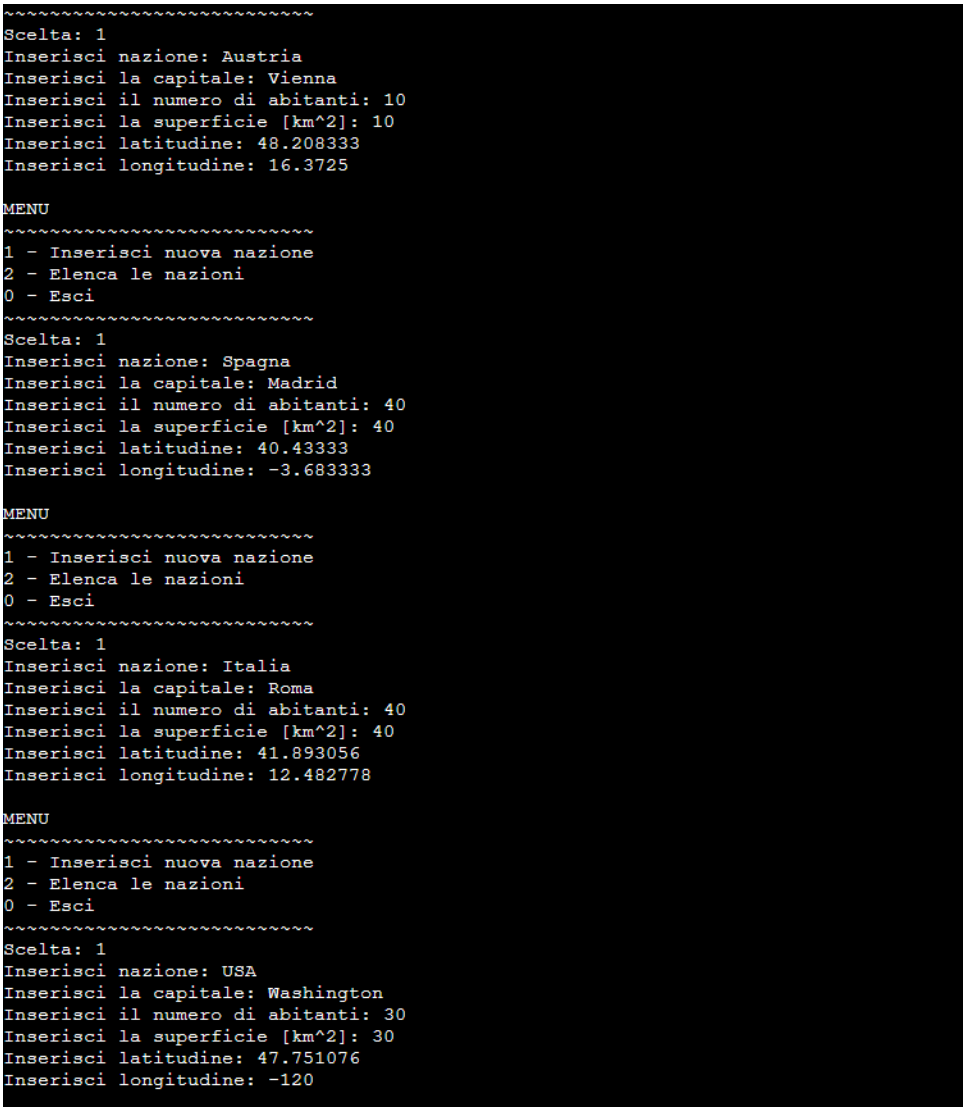

Figura 5: Ordinamento per longitudine, 2.

Figura 6: Ordinamento per longitudine, 3.

5. RIFERIMENTI

- Per la struttura dell'albero mi sono ispirato a quanto fatto a lezione.
- Per la funzione `str2STR` ho consultato i due siti: `https://www.delftstack.com/it/howto/cpp/how-to-convert-string-to-uppercase-cpp/`, `https://en.cppreference.com/w/cpp/algorithm/transform` (ultimo accesso 21/12/2022).