Programmazione 2
e Laboratorio di Programmazione

Corso di Laurea in

Informatica

Università degli Studi di Napoli “Parthenope”

Anno Accademico 2023-2024

Prof. Luigi Catuogno

1

Esercizi svolti

2

Riepilogo su array e puntatori

3

Esercizio: la morra cinese

La Morra Cinese è un gioco antichissimo già attestato in Cina ai tempi della Dinastia Han (206 AC – 220 DC). Se ne contano innumerevoli varianti ed è stato oggetto di centinaia di studi scientifici (delle più varie discipline).

Scriviamo un programma C++ che implementa una versione in cui un giocatore umano affronta il calcolatore per un numero di round fissato all'inizio. Al giocatore che prevale in ciascun round, viene assegnato un punto. Nessun punteggio viene attribuito in caso di parità. Completati tutti i round, il programma visualizza i punteggi e termina.

4

Esercizio: la morra cinese

```
 1  #include<iostream>
 2  #include<cstdlib>
 3  #define NUMGESTI 3
 4  using namespace std;
 5
 6  const int SASSO=0,CARTA=1,FORBICI=2;
 7
 8  bool batte[3][3]{{false,false,true},
 9                   {true,false,false},
10                   {false,true,false}};
11
12  char nomeG[3][8]={"sasso","carta","forbici"};
```

5

Esercizio: la morra cinese

```
13  int main()
14  {
15          int seed, round;
16          int mia_mossa, sua_mossa, miei_punti=0, suoi_punti=0;
16
17          cout << "*** Morra Cinese ***"<<endl;
18          cout << "Inserisci il seed: ";
19          cin >> seed;
20          srand(seed);
21          cout << "Inserisci il numero di round: ";
22          cin >> round;
```

6

Esercizio: la morra cinese

```
23  for(int i=0;i<round;i++) {
24                  sua_mossa=rand()%NUMGESTI;
25                  cout<<"La mia mossa(0=Sasso,1=Carta,2=Forbici) è: ";
26                  cin>> mia_mossa;
27                  cout<<"La sua mossa è "<< nomeG[sua_mossa];
28                  if (batte[mia_mossa][sua_mossa]) {
29                          miei_punti++;
30                          cout << " Vinco io"<<endl;
31                  } else if (batte[sua_mossa][mia_mossa]) {
32                          suoi_punti++;
33                          cout << " Vince lui"<<endl;
34                  } else
35                          cout << " Pari!"<<endl;
36          }
37          cout << "Io: "<<miei_punti<<", Lui: "<<suoi_punti<<endl;
38  }
```

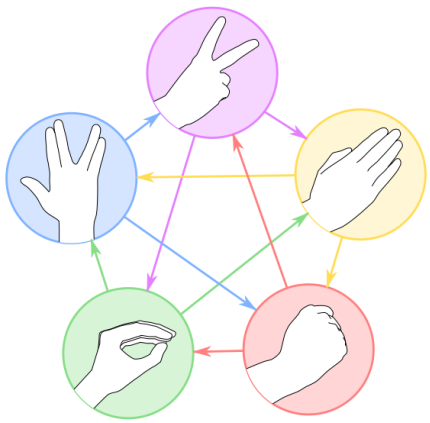
7

Esercizio: la morra cinese

- Una variante della morra cinese, è stata inventata da Karen Bryla e Sam Kass negli anni 2000.
  - Alla morra tradizionale sono stati aggiunti la lucertola e Spock. Le relazioni tra i simboli sono riassunte di seguito e nel grafico al lato

Forbici tagliano carta; carta avvolge sasso; sasso rompe forbici; forbici decapitano lucertola; lucertola mangia carta; carta smentisce Spock; Spock vaporizza sasso; sasso schiaccia lucertola; lucertola avvelena Spock; Spock rompe forbici.

http://www.samkass.com/theories/RPSSL.html



8

Esercizio: *numeri uguali in posti uguali*

Scrivere un programma C++ che chieda all'utente di riempire due array di 5 interi calcoli quante volte, nei due array compare lo stesso numero nella stessa posizione.

Per esempio se **`arr1={1,12,54,79,39}`** e **`arr2={1,79,47,81,39}`** il risultato è 2 (1 e39)

Il conteggio deve essere effettuata da una funzione che prende il puntatore ai due array e la loro lunghezza e restituisca un intero.

Nella funzione, lo scorrimento dell'array deve essere effettuato utilizzando l'aritmetica dei puntatori

9

Esercizio: *numeri uguali in posti uguali*

```
 4  int contauguali(int *p1, int *p2, int len)
 5  {
 6          int cnt=0;
 7          for (int i=0;i<len;i++){
 8                  if (*p1==*p2)
 9                          cnt++;
10                  p1++;
11                  p2++;
12          }
13          return cnt;
14  }
```

Le variabili puntatore sono variabili locali (il dato puntato non lo è). Queste espressioni hanno effetto soltanto all'interno della funzione.

10

Esercizio: *al contrario*

Scrivere un programma C++ che chieda all'utente di riempire un array di 5 interi. Al termine dell'input, gli elementi dell'array devono essere messi al contrario.

Per esempio l'utente inserisce **`arr1={1,12,54,79,39}`** al termine del programma, deve risultare che **`arr1={39,79,54,12,1}`**

Serve una funzione **`swap`** che scambi due elementi (passaggio dei parametri per indirizzo)

Nel programma, lo scorrimento dell'array deve essere effettuato utilizzando l'aritmetica dei puntatori

11

Esercizio: *al contrario*

```
1  #include<iostream>
2  using namespace std;
3
4  void swap(int *v1, int *v2)
5  {
6          int t;
7          t=*v1; *v1=*v2; *v2=t;
8  }
9
```

12

Esercizio: *al contrario*

```
10 void inverti_array(int *array, int len)
11 {
12         int *sinistra, *destra;
13
14         sinistra=array;
15         destra=array+(len-1);
16
17         while(sinistra<destra){
18                 swap(sinistra,destra);
19                 sinistra++;
20                 destra--;
21         }
22 }
23
```

13

Esercizio: *al contrario*

```
24 int main()
25 {
26         int sequenza[5]={1,2,3,4,5};
27
28         inverti_array(sequenza,5);
29
30         cout <<"( ";
31         for(int i=0;i<5;i++)
32                 cout << sequenza[i]<< " ";
33         cout<<")"<<endl;
34 }
```

14

Esercizio: *sequenze palindrome?*

Si scriva un programma in C++ che chieda all'utente di inserire una sequenza di interi in un array di 8 elementi e dica se tale sequenza è *palindroma.*

15

Esercizio: *sequenze palindrome?*

```
1   #include<iostream>
2
3   using namespace std;
4
5   int main()
6   {
7           int S[8], i, j;
8           bool palindroma=true;
9
10          cout << "Inserisci la sequenza di 10 numeri: "<< endl;
11          for (int k=0;k<8;k++){
12                  cout << "S["<<k<<"]=";
13                  cin >> S[k];
14          }
…   …
```

Siamo ottimisti, all'inizio stimiamo che la sequenza sia palindroma. Poi controlliamo e nel caso, modifichiamo questo valore.

16

Esercizio: *sequenze palindrome?*

```
1   #include<iostream>
2
3   using namespace std;
4
5   int main()
6   {
7         int S[8], i, j;
8         bool palindroma=true;
9
10        cout << "Inserisci la sequenza di 10 numeri: "<< endl;
11        for (int k=0;k<8;k++){
12              cout << "S["<<k<<"]=";
13              cin >> S[k];
14        }
…   …
```

I valori in un array devono essere inseriti uno alla volta (non ci sono assegnamenti tra array). Il metodo classico è inserire l'istruzione di input/assegnamento all'interno di un ciclo.

17

Esercizio: *sequenze palindrome?*

```
15        i=0;
16        j=7;
17
18        while (i<j){
19              if (S[i]!=S[j]) {
20                    palindroma=false;
21                    break;
22              }
23              i++;
24              j--;
25        }
26
…   …
```

i e **j** contengono la posizione dei due elementi «speculari» dell'array che devono essere confrontati tra loro. Inizialmente contengono rispettivamente il primo e l'ultimo elemento.

La coppia di elementi è confrontata e nel caso in cui questi risultassero diversi… **palindroma** passa a **false** e l'analisi termina (uscendo dal ciclo)

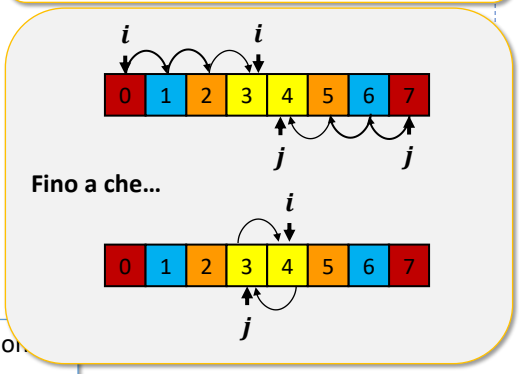
Se invece il confronto ha successo, si passa a selezionare la successiva coppia da prendere in esame.

18

Esercizio: *sequenze palindrome?*

```
15          i=0;
16          j=7;
17
18          while (i<j){
19                  if (S[i]!=S[j]) {
20                          palindroma=false;
21                          break;
22                  }
23                  i++;
24                  j--;
25          }
26
…   …
```

Si continua così, confrontando il primo e l'ultimo elemento, il secondo e il penultimo…



Se invece il confronto ha successo, si passa a selezion… la successiva coppia da prendere in esame.

19

Esercizio: *sequenze palindrome?*

```
…           …
27          cout << "S ";
28          if (!palindroma)
29                  cout << "non ";
30          cout << "e' palindroma!"<<endl;
31  }
```

20

Esercizio: *sequenze palindrome coi puntatori*

Scrivere un programma C++ che chieda all'utente di riempire un array di 5 interi e verifichi se la sequenza inserita è palindroma. Nello svolgimento:

La verifica deve essere effettuata dalla funzione:

```
bool is_palindrome(int *sequence, int len)
```

che prende il puntatore all'array e la sua lunghezza e restituisce true se la sequenza è palindroma, false altrimenti.

Nella funzione, lo scorrimento dell'array deve essere effettuato con dei puntatori.

21

Esercizio: *sequenze palindrome coi puntatori*

```
23 int main()
24 {
25         const int MAX=5;
26         int main_array[MAX]={1,4,2,4,1};
27
28         cout << "isPalindrome=" << is_palindrome(main_array,MAX);
29         cout << endl;
30 }
```

22

Esercizio: *sequenze palindrome coi puntatori*

```
4  bool is_palindrome(int *seq, int size){
5          int *i, *j;
6
7          i=seq;
8          j=seq+(size-1);
9
10         while(i<j){
11                 if (*i!=*j)
12                         return false;
13                 i++;
14                 j--;
15         }
16         return true;
17 }
```

`i` punta all'indirizzo base dell'array (prima cella)

`j` punta all'ultima cella dell'array

`i` passa alla cella successiva

`j` passa alla cella precedente

23