Programmazione 2 e Lab. di programmazione 2

Corso di Laurea in Informatica - Anno Accademico 2022-23

Docenti
Prof. Angelo Ciaramella
[angelo.ciaramella@uniparthenope.it]
Prof. Luigi Catuogno
[luigi.catuogno@uniparthenope.it]
Tutor
Dott. Antonio Vanzanella
[antonio.vanzanella@studenti.uniparthenope.it]

1

Orari e modalità di ricevimento studenti

Docenti:

Prof. Angelo Ciaramella	Martedì dalle 14:00 alle 16:00 - telematico (codice Teams r3p3w0z)
Prof. Luigi Catuogno	Giovedì dalle 11:00 alle 13:00 - telematico (codice Teams)

Tutor:

Dott. Antonio Vanzanella	Martedì dalle 11:00 alle 13:00 - telematico (codice Teams 92dbag0)

4

Il Linguaggio C++

(per programmatori C)

Parte prima

5

Tipo di dati **`bool`**

6

Tipo di dati **bool**

```
1  // Programma che illustra il tipo 'bool' del C++
2
3  #include <iostream>
4  using namespace std;
5  int main()
6  {
7        bool a1=true, b1=false, x;
8        x = a1 && b1;
9        cout << "a1 AND b1=" << x << endl;
10 }
```

```
a1 AND b1=0
```

7

Tipo di dati **bool**

```
7        bool a1=true, b1=false;
```

Le variabili di tipo **bool** possono essere inizializzate utilizzando in maniera equivalente gli interi **0** e **1** e le costanti **false** e **true**;

```
8        x = a1 && b1;
```

Può essere loro assegnato il risultato di una espressione logico-relazionale.

8

Tipo di dati **bool**

```
9        cout << "a1 AND b1=" << x << endl;
```

Generalmente una variabile **bool** è memorizzata come una variabile intera(*) (sebbene non possa avere valori diversi da 0 e 1) e come tale è visualizzata.

(*) un byte con un solo 1 bit significativo.

9

Tipo di dati **bool**

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6         bool a1, b1, x;
7         cout << "inserire a1 e b1: ";
8         cin >> a1 >> b1;
9         x = a1 && b1;
10        cout << "a1 AND b1= " << boolalpha << x << endl;
11 }
```

Il manipolatore **boolalpha** determina la traduzione tra le costanti numeriche e quelle testuali dei valori booleani

```
inserire a1 e b1: 1 0
a1 AND b1= false
```

10

Tipo di dati **bool**

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6         bool a1, b1, x;
7         cout << "inserire a1 e b1: ";
8         cin >> boolalpha >> a1 >> b1;
9         x = a1 && b1;
10        cout << "a1 AND b1= " << boolalpha << x << endl;
11 }
```

Il manipolatore **boolalpha** determina la traduzione tra le costanti numeriche e quelle testuali dei valori booleani.

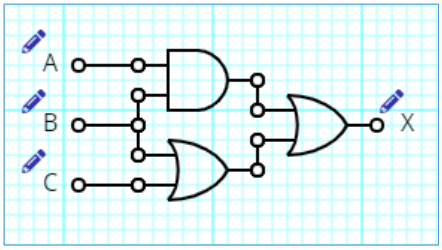
```
inserire a1 e b1: true false
a1 AND b1= false
```

11

Esercizio: *una funzione booleana*

Si scriva un programma C++ che chieda all'utente fornire in input i valori da applicare ai tre ingressi A,B e C del circuito al lato (codificate con delle variabili booleane), quindi calcoli e visualizzi il valore dell'uscita X.

1) Input e output devono essere effettuate utilizzando le costanti testuali true e false
2) Il circuito deve essere implementato nella funzione **circuito()**, mentre la funzione **main()** si dovrà occupare solo di chiedere l'input, invocare la funzione e visualizzarne il risultato.



12

Esercizio: *una funzione booleana*

```
1  #include<iostream>
2  using namespace std;
3
4  bool circuito (bool a, bool b, bool c)
5  {
6         return (a&&b)||(b||c);
7  }
8
```

13

Esercizio: *una funzione booleana*

```
9  int main()
10 {
11        bool A, B, C, X;
12
13        cout << "Inserisci A,B e C: ";
14        cin >> boolalpha >> A >> B >> C;
15        X=circuito(A,B,C);
16        cout << "X=" << boolalpha << X << endl;
17 }
```

14

Ripasso su Array e Matrici

15

Ripasso su Array e Matrici

Struttura dati composta da un insieme di elementi dello stesso tipo

Ciascun elemento è identificato univocamente mediante un indice (un numero intero che ne definisce la posizione nell'array)

La lunghezza di un array è stabilita al **momento** della sua dichiarazione (e non cambia)

In un array lungo N, troveranno posto N elementi del tipo base che saranno identificati con gli indici da 0 a n-1. Indicare un indice al di fuori di questo intervallo è un errore *(non sempre segnalato dal compilatore…)*

17

Ripasso su Array e Matrici

Dichiarazione di un array:

tipo_base ***identificatore_array*** **[***dimensione(int)***]**;

```
int serie[10];
float v1[4],v2[4];
bool x[4];
char c[12];
```

18

Ripasso su Array e Matrici

Dichiarazione di un array:

tipo_base ***identificatore_array*** **[***dimensione(int)***]**;

```
int serie[10];
float v1[4],v2[4];
bool x[3],z;
char c[12];
```

19

Ripasso su Array e Matrici

Espressioni con gli elementi di un array:

```
x[0]=true;
x[1]=false;
x[2]=true;

z=x[0] && x[1] && x[2];
```

20

Ripasso su Array e Matrici

L'indice può anche essere il risultato di una espressione *a valori interi* purché sia garantito che il risultato indichi una posizione compresa tra 0 e N-1 (dove N è la lunghezza dell'array)

```
for(int i=0;i<10;i++)
     cin >> serie[i];
```

21

Ripasso su Array e Matrici

Si possono avere array *multidimensionali:*

tipo identificatore **[***dim*$_1$**][***dim*$_2$**] … [***dim*$_n$**];**

```
int tabella[12][2];
float m1[4][4],m2[4][5][6];
```

In un array a N dimensioni, ogni elemento è identificato da una N-pla di indici che ne rappresentano le «coordinate» all'interno dell'array.

22

Ripasso su Array e Matrici

In memoria, gli elementi di un array occupano uno spazio contiguo.

```
int vect[3];
```

vect[0]
vect[1]
vect[2]

23

Ripasso su Array e Matrici

In memoria, gli elementi di un array occupano uno spazio contiguo.

```
int matr[2][2];
```

matr[0][0]
matr[0][1]
matr[1][0]
Matr[1][1]

24

Ripasso su Array e Matrici

Dichiarazione con inizializzazione

```
int vect[3]={0,1,0};
float m1[2][3]= {{1,3,4},{4,5,6}};
float m2[2][3]= {1,2,3,4,5,6};
```

Gli array sono memorizzati una riga alla volta, pertanto m1 e m2 sono inizializzate con i medesimi valori

25

Ripasso su Array e Matrici

Dichiarazione con inizializzazione

```
int tabella[12][2]={0};
```

Se nell'inizializzazione si specifica il valore di un numero di elementi minore delle posizioni dichiarate, tutti gli elementi rimanenti saranno inizializzati a zero.

Questa dichiarazione, è un modo compatto per inizializzare una matrice con tutti i suoi elementi a zero.

26

Esempio: operazioni con i vettori

Prodotto *scalare* di due vettori di 3 elementi:

$$s = v \cdot w = \sum_{i=0}^{2} v_i w_i$$

27

Esempio: operazioni con i vettori

Prodotto *scalare* di due vettori di 3 elementi:

$$s = v \cdot w = \sum_{i=0}^{2} v_i w_i$$

```
float s=0, v[3], w[3];
…
s=(v[0]*w[0] +  v[1]*w[1]+ v[2]*w[2]);
```

28

Esempio: operazioni con i vettori

```
int main()
{
     float v[3]={1,1,6}, w[3]={2,1,-1};
     float s=0;
     cout << "Prodotto scalare:"<<endl;
     cout << "(";
     for (int i=0;i<3;i++) {
          cout << v[i] <<",";
          s=s+v[i]*w[i];
     }
…
```

29

Esempio: operazioni con i vettori

```
…
      cout << ") . (";
      for (int i=0;i<3;i++)
            cout << w[i] <<",";
      cout << ") = "<< s << endl;
}
```

30

Esempio: operazioni con i vettori

```
Prodotto scalare:
(1,1,6,) . (2,1,-1,) = -3
```

31

Esempio: *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.*

32

Esempio: *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.

33

Esempio: *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 1      ri: "<< 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.

34

Esempio: *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)

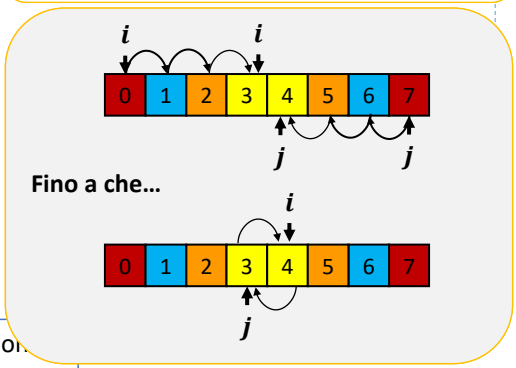
Se invece il confronto ha successo, si passa a selezionare la successiva coppia da prendere in esame.

35

Esempio: *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.

36

Esempio: *sequenze palindrome?*

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

37

Esercizio: *il Crivello di Eratostene*

Il Crivello di Eratostene

Antico metodo per la ricerca dei numeri primi attribuito al matematico greco Eratostene da Cirene, vissuto tra il terzo e il secondo secolo a.C.

38

Esercizio: *il Crivello di Eratostene*

- Fissato n come limite superiore:
- Si utilizza un elenco (setaccio) dei numeri da 2 a n
  - Si scorre l'elenco partendo da 2 e si prendono in esame uno alla volta, tutti i numeri che non siano stati già «cancellati»
  - si «cancellano» tutti i multipli del numero in esame (escluso il numero stesso);
  - al termine del procedimento, i numeri primi sono quelli che non risultano cancellati;

39

Esercizio: *il Crivello di Eratostene*

$N = 30;$

Cominciamo da **2**...

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

40

Esercizio: *il Crivello di Eratostene*

$N = 30;$

Cominciamo da **2**...

	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	22	23	24	25	26	27	28	29	30

	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~

41

Esercizio: *il Crivello di Eratostene*

$N = 30$;

Continuiamo con **3**…

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~	~~28~~	29	~~30~~

42

Esercizio: *il Crivello di Eratostene*

$N = 30$;

Saltato il **4** (già cancellato) si passa al **5**…

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~	~~28~~	29	~~30~~

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~

43

Esercizio: *il Crivello di Eratostene*

$N = 30;$

I numeri che non sono stati cancellati sono numeri primi…

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~

44

Esercizio: *il Crivello di Eratostene*

Si scriva un programma in C++ che implementi il Crivello di Eratostene con N=100.

```
I numeri primi da 1 a 100 sono:
2, 3, 5, 7, 11, 13, 17, 19, 23,
29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 71, 73, 79, 83, 89, 97
```

Suggerimento: per il crivello, si utilizzi un array di N elementi di tipo **`bool`**.

45

Esercizio: *il Crivello di Eratostene*

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6         bool mat[101]={false};
7
8         cout << "I numeri primi da 2 a 100 sono: ";
9         for (int i=2;i<=100;i++){
10              if (mat[i])
11                    continue;
12              for (int j=2*i;j<=100;j+=i)
13                    mat[j]=true;
14        }
```

Per «cancellare» il numero **j** si pone **mat[j]=true**

Di ciascun numero da 2 a 100,

che non è stato ancora cancellato,

si cancellano tutti i multipli dal crivello

46

Esercizio: *il Crivello di Eratostene*

```
15        cout << 2;
16        for (int i=3;i<=100;i++)
17              if (!mat[i])
18                    cout << ", " << i ;
19        cout <<endl;
20 }
```

Si visualizzano tutti i numeri non cancellati

47

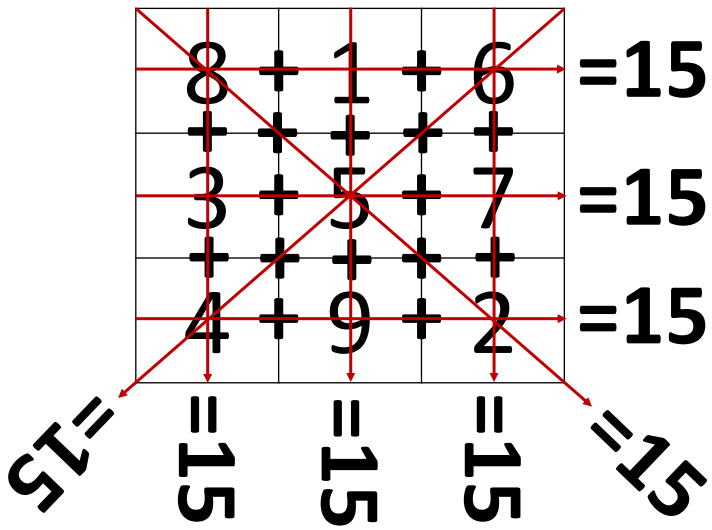
Esercizio: *Il quadrato magico…*

Un quadrato magico è una matrice di interi positivi $N \times N$ riempito con i numeri da 1 a N^2 disposti in modo tale che la somma dei numeri su occupano ciascuna riga, o colonna o diagonale, dia sempre lo stesso valore. Tale valore è costante e dipende dall'ordine del quadrato.

8	1	6
3	5	7
4	9	2

Quadrato magico di ordine 3.
Il numero magico è 15

48



49

Esercizio: *Il quadrato magico…*

- Si scriva il programma C++ che:
  - Chieda all'utente di riempire una matrice di dimensioni 3x3 con i numeri da 1 a 9 (scelti ciascuno una sola volta)
  - Dica se la matrice inserita è un quadrato magico;

50

Esercizio: *Il quadrato magico…*

4	9	2
8	1	6
3	5	7

NO

2	9	4
7	5	3
6	1	8

SI

51