

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int i;
    char s[MAXPAROLA]; // vettore di carattere
    // della lunghezza della lunghezza della parola
    char s2[MAXRIGA];
    int i2, len, lunghezza;
    int i3;

    printf("MAXPAROLA: %d\n", MAXPAROLA);
    printf("MAXRIGA: %d\n", MAXRIGA);

    printf("Inserisci una parola: ");
    fgets(s, MAXPAROLA, stdin);

    len = strlen(s);
    printf("La parola %s ha %d caratteri\n", s, len);

    printf("Inserisci una riga: ");
    fgets(s2, MAXRIGA, stdin);

    printf("La riga %s ha %d caratteri\n", s2, strlen(s2));

    return 0;
}
```

Programmazione in C

Unità Cicli ed iterazioni

```
(argc = 3)
1
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
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Cicli ed iterazioni

- La ripetizione
- Istruzione while
- Schemi ricorrenti nei cicli
- Istruzione for
- Approfondimenti
- Esercizi proposti
- Sommario

```
(argc = 3)
1
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
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Riferimenti al materiale

- Testi
 - Kernighan & Ritchie: capitolo 3
 - Cabodi, Quer, Sonza Reorda: capitolo 4
 - Dietel & Dietel: capitolo 4
- Dispense
 - Scheda: "Cicli ed iterazioni in C"

```

#include <bits/stdc++.h>
#include <string.h>
#include <ctype.h>

using namespace std;

int main() {
    string s;
    int n;
    while (cin >> s && cin >> n) {
        if (s.length() < n) {
            cout << "Error: string length is less than n" << endl;
            continue;
        }
        string t(s.substr(0, n));
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (t[i] == s[i]) {
                count++;
            }
        }
        cout << count << endl;
    }
}

```

Cicli ed iterazioni

La ripetizione

```

#include <bits/stdc++.h>
#include <string.h>
#include <ctype.h>

using namespace std;

int main() {
    string s;
    int n;
    while (cin >> s && cin >> n) {
        if (s.length() < n) {
            cout << "Error: string length is less than n" << endl;
            continue;
        }
        string t(s.substr(0, n));
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (t[i] == s[i]) {
                count++;
            }
        }
        cout << count << endl;
    }
}

```

La ripetizione

Concetto di ciclo

La ripetizione

- Concetto di ciclo
- Struttura di un ciclo
- Numero di iterazioni note
- Numero di iterazioni ignote

Flusso di esecuzione ciclico

- È spesso utile poter **ripetere** alcune parti del programma più volte
- Nel diagramma di flusso, corrisponde a "tornare indietro" ad un blocco precedente
- Solitamente la ripetizione è controllata da una condizione booleana

```

graph TD
    A[A] --> B[B]
    B --> C[C]
    C --> D{D?}
    D -- V --> B
    D -- F --> E[E]

```

Flusso di esecuzione ciclico

```

graph TD
    A[A] --> B[B]
    B --> C[C]
    C --> D{D?}
    D -- V --> B
    D -- F --> E[E]

```

Errore frequente

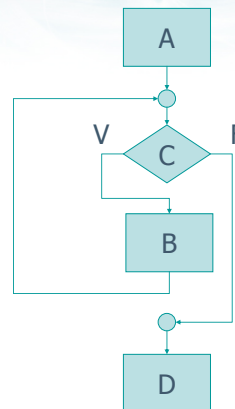
- Ogni ciclo porta in sé il rischio di un grave errore di programmazione: il fatto che il ciclo venga ripetuto indefinitamente, senza mai uscire
- Il programmatore deve garantire che ogni ciclo, dopo un certo numero di iterazioni, venga terminato
 - La condizione booleana di controllo dell'iterazione **deve** divenire falsa

Istruzioni eseguibili ed eseguite

- Istruzioni eseguibili
 - Le istruzioni che fanno parte del programma
 - Corrispondono alle istruzioni del sorgente C
- Istruzioni eseguite
 - Le istruzioni effettivamente eseguite durante una specifica esecuzione del programma
 - Dipendono dai dati inseriti
- Nel caso di scelte, alcune istruzioni eseguibili non verranno eseguite
- Nel caso di cicli, alcune istruzioni eseguibili verranno eseguite varie volte

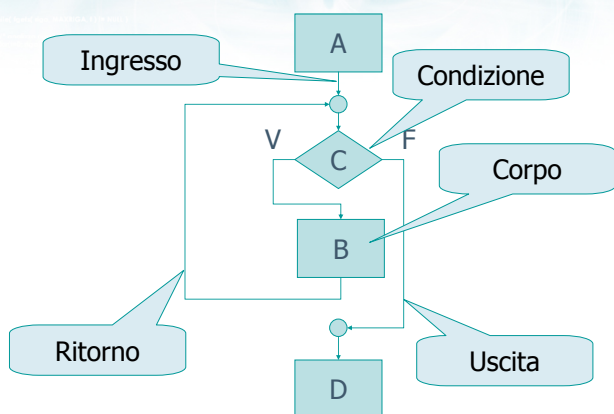
10

Notazione grafica (while)



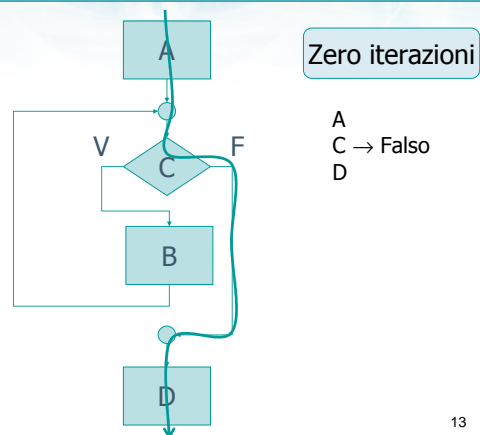
11

Notazione grafica (while)



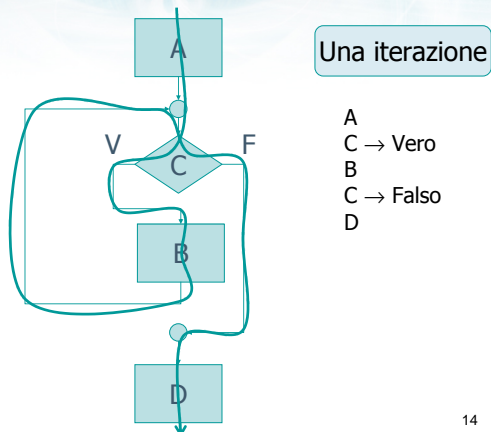
12

Flussi di esecuzione



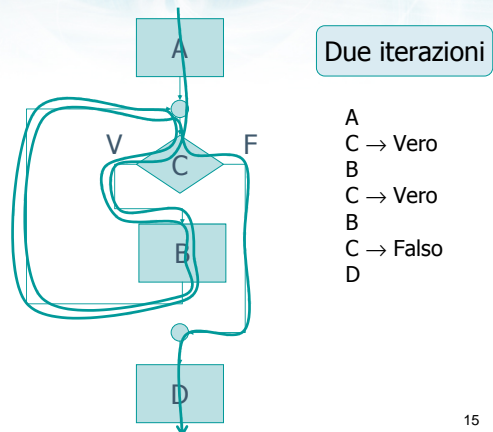
13

Flussi di esecuzione



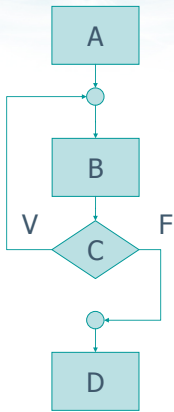
14

Flussi di esecuzione



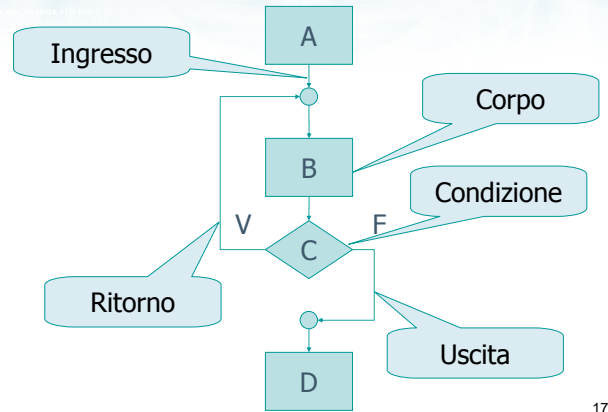
15

Notazione grafica (do-while)



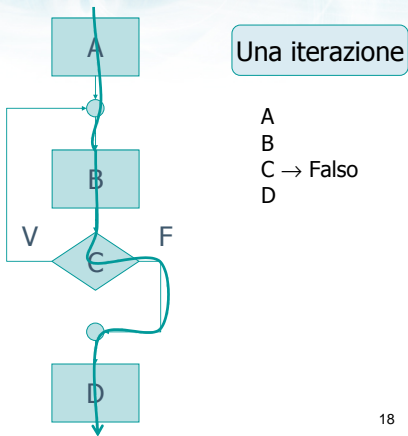
16

Notazione grafica (do-while)



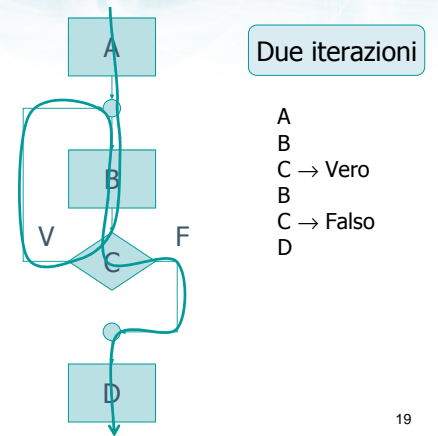
17

Flussi di esecuzione



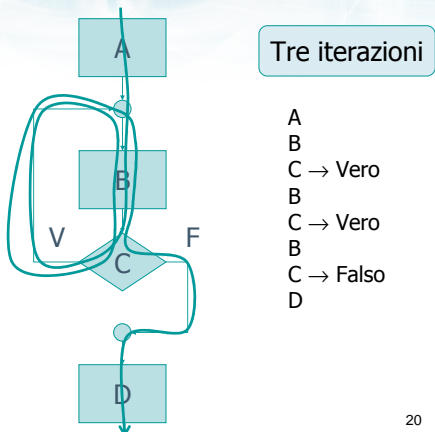
18

Flussi di esecuzione



19

Flussi di esecuzione



20

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXFASOLA 30
#define MAXDGA 80

int main(int argc, char *argv[])
{
    int fasola(MAXFASOLA); /* valore di controllo
    della lunghezza della stringa della parola */
    char *fasola(MAXFASOLA);
    int i, n;
    FILE *f;

    printf("MAXFASOLA: %d\n", fasola);

    /* leggi la stringa */
    printf("Inserisci la parola (max %d caratteri)\n", fasola);
    scanf("%s", fasola);
    printf("La parola è: %s\n", fasola);

    /* verifica se la parola è palindroma */
    printf("La parola %s è palindroma? (s/n)\n", fasola);
    scanf(" %c", &n);
    while (getchar() != '\n')
        continue;
  
```

La ripetizione

Struttura di un ciclo

Problemi

- ▶ Nello strutturare un ciclo occorre garantire:
 - Che il ciclo possa terminare
 - Che il numero di iterazioni sia quello desiderato
- ▶ Il corpo centrale del ciclo può venire eseguito più volte:
 - La prima volta lavorerà con variabili che sono state inizializzate al di fuori del ciclo
 - Le volte successive lavorerà con variabili che possono essere state modificate nell'iterazione precedente
 - Garantire la correttezza sia della prima, che delle altre iterazioni

22

Anatomia di un ciclo (1/5)

- ▶ Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento

23

Anatomia di un ciclo (2/5)

- ▶ Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nel corpo o nella condizione)
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento

24

Anatomia di un ciclo (3/5)

- ▶ Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa
 - Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento)
 - Corpo
 - Aggiornamento

25

Anatomia di un ciclo (4/5)

- ▶ Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Le istruzioni che effettivamente occorre ripetere
 - Sono lo scopo per cui il ciclo viene realizzato
 - Posso usare le variabili inizializzate
 - Posso modificare le variabili
 - Aggiornamento

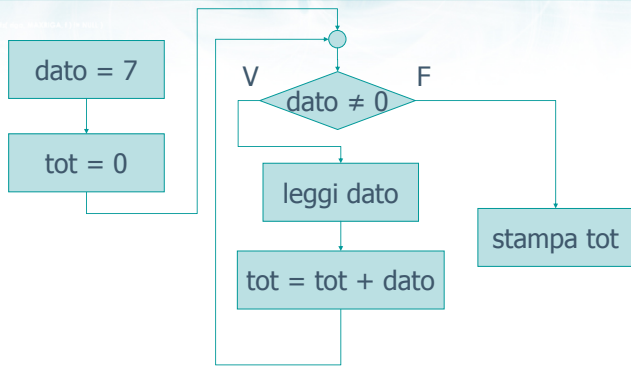
26

Anatomia di un ciclo (5/5)

- ▶ Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento
 - Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione
 - Tengono "traccia" del progresso dell'iterazione

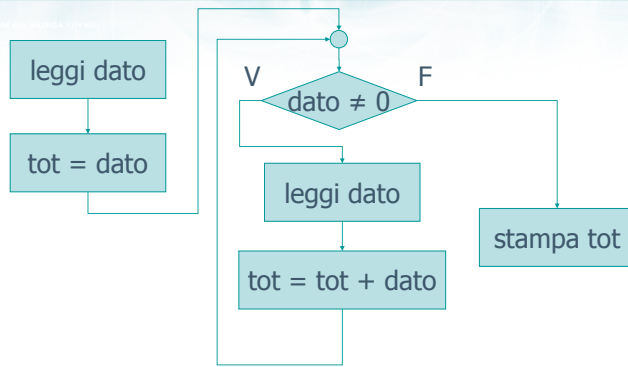
27

Soluzione 1



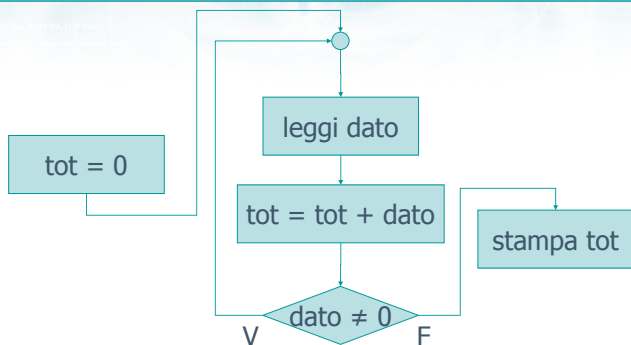
40

Soluzione 2



41

Soluzione 3

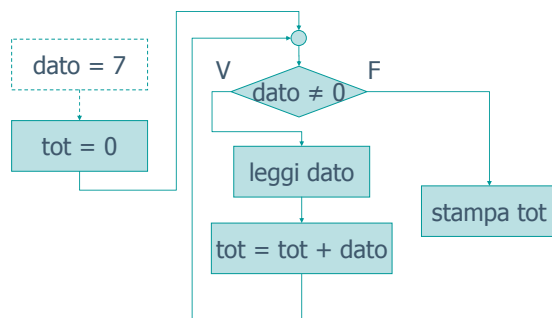


42



Errore frequente

- Dimenticare l'inizializzazione di una variabile utilizzata all'interno del ciclo

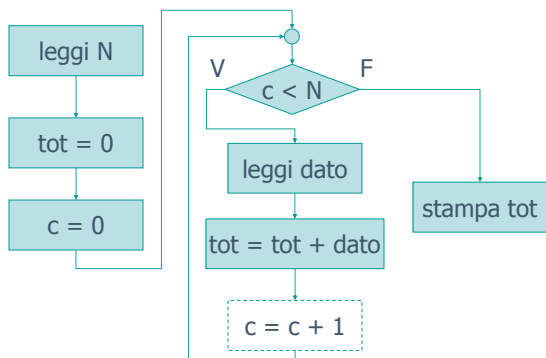


43



Errore frequente

- Dimenticare l'incremento della variabile contatore

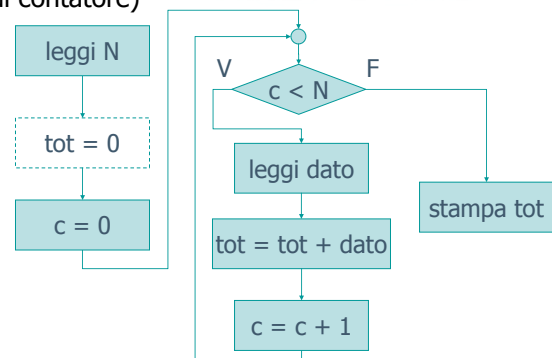


44



Errore frequente

- Dimenticare di inizializzare le altre variabili (oltre al contatore)



45


```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAZZAROLA]; // vettore di contatori
    // della frequenza delle lunghezze delle parole
    char parola[MAZZAROLA];
    int i, len, lunghezza;
    int r;

    for(i=0; i<MAZZAROLA; i++)
        freq[i]=0;

    while (argc > 1)
    {
        printf("Inserisci una parola (o 'q' per uscire): ");
        fgets(parola, MAZZAROLA, stdin);
        len = strlen(parola);
        if (parola[len-1] != '\n')
            continue;
        for(i=0; i<len; i++)
            freq[parola[i]]++;
    }

    while (argc > 1)
        printf("%s: %d\n", argv[1], freq[argv[1]]);
}

```

Cicli ed iterazioni

Istruzione while

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAZZAROLA]; // vettore di contatori
    // della frequenza delle lunghezze delle parole
    char parola[MAZZAROLA];
    int i, len, lunghezza;
    int r;

    for(i=0; i<MAZZAROLA; i++)
        freq[i]=0;

    while (argc > 1)
    {
        printf("Inserisci una parola (o 'q' per uscire): ");
        fgets(parola, MAZZAROLA, stdin);
        len = strlen(parola);
        if (parola[len-1] != '\n')
            continue;
        for(i=0; i<len; i++)
            freq[parola[i]]++;
    }

    while (argc > 1)
        printf("%s: %d\n", argv[1], freq[argv[1]]);
}

```

Istruzione while

Sintassi dell'istruzione

```

while ( C )
{
    B ;
}

```

```

graph TD
    Start(( )) --> C{C}
    C -- V --> B[B]
    B --> C
    C -- F --> End(( ))

```

```

while ( C )
{
    B ;
}

```

Istruzione while

- Sintassi dell'istruzione
- Esercizio "Media aritmetica"
- Esecuzione del programma
- Cicli while annidati
- Esercizio "Quadrato"

```

while ( C )
{
    B ;
}

```

Istruzioni di ripetizione in C

- Nel linguaggio C esistono tre distinte istruzioni di iterazione
 - while
 - do-while
 - for
- La forma più generale è l'istruzione di tipo while
- L'istruzione do-while si usa in taluni contesti (es. controllo errori di input)
- L'istruzione for è usatissima, soprattutto per numero di iterazioni noto

```

while ( C )
{
    B ;
}

```

Comportamento del while

1. Valuta la condizione C
2. Se C è falsa, salta completamente l'iterazione e vai all'istruzione che segue la }
3. Se C è vera, esegui una volta il blocco di istruzioni B
4. Al termine del blocco B, ritorna al punto 1. per rivalutare la condizione C

Numero di iterazioni note

```
int i, N ;

i = 0 ;
while ( i < N )
{
    /* Corpo dell'iterazione */
    ...

    i = i + 1 ;
}
```

7

Esempio

```
int i ;

i = 1 ;
while ( i <= 10 )
{
    printf("Numero = %d\n", i) ;
    i = i + 1 ;
}
```



num1-10.c

8

Esempio

```
int i, n ;
float f ;
.... /* leggi n */ ....
i = 2 ;
f = 1.0 ;
while ( i <= n )
{
    f = f * i ;
    i = i + 1 ;
}
printf("Fattoriale di %d = %f\n",
n, f);
```



fatt.c

9

Particolarità

- ▶ Nel caso in cui il corpo del `while` sia composto di una sola istruzione, si possono omettere le parentesi graffe
 - Non succede quasi mai

```
while ( C )
{
    B ;
}
```

```
while ( C )
    B ;
```



Istruzione `while`

Esercizio "Media aritmetica"

Esercizio "Media aritmetica"

- ▶ Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Leggere n numeri reali
 - Calcolare e visualizzare la media aritmetica di tali numeri
- ▶ Osservazione
 - Attenzione al caso in cui $n \leq 0$

12

```

Prompt dei comandi

MEDIA ARITMETICA

Introduci n: 3
Ora introduci 3 valori
Valore 1: 6.5
Valore 2: 2.5
Valore 3: 3.0

Risultato: 4.000000

```

13

- Acquisisci n
- Inizializza totale = 0
- Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
- Calcola e stampa la media = totale / n

14

- Acquisisci n
- Se n > 0
 - Inizializza totale = 0
 - Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
 - Calcola e stampa la media = totale / n
- Altrimenti stampa messaggio di errore

15

```

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

int main(void)
{
    int i, n ;
    float dato ;
    float somma ;

    printf("MEDIA ARITMETICA\n");

    /* Leggi n */
    printf("Introduci n: ");
    scanf("%d", &n) ;

```

16

```

/* Controlla la correttezza del valore n */
if( n>0 )
{
    /* n corretto... procedi! */
    ...vedi lucido seguente....
}
else
{
    /* n errato in quanto e' n <= 0 */
    printf("Non ci sono dati da inserire\n");
    printf("Impossibile calcolare la media\n");
}
} /* main */

```

17

```

/* Leggi i valori e calcola la media */
printf("Ora immetti %d valori\n", n) ;

somma = 0.0 ;
i = 0 ;
while( i < n )
{
    printf("valore %d: ", i+1) ;
    scanf("%f", &dato) ;

    somma = somma + dato ;

    i = i + 1 ;
}

printf("Risultato: %f\n", somma/n) ;

```

18

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq(MAXFREQ); // vettore di contatori
    della frequenza delle lunghezze delle parole
    char digit[MAXDIGIT];
    int i, len, lunghezza;
    int n;

    for(i=0; i<MAXFREQ; i++)
        freq[i]=0;

    while(n=scanf("%d", &n))
    {
        if(n>0)
        {
            for(i=0; i<n; i++)
            {
                char s[100];
                scanf("%s", s);
                len=strlen(s);
                freq[len]++;
            }
        }
        else
            break;
    }

    printf("Lunghezza delle parole\n");
    for(i=1; i<MAXFREQ; i++)
        printf("%d: %d\n", i, freq[i]);
}

```

Istruzione while

Esecuzione del programma

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq(MAXFREQ); // vettore di contatori
    della frequenza delle lunghezze delle parole
    char digit[MAXDIGIT];
    int i, len, lunghezza;
    int n;

    for(i=0; i<MAXFREQ; i++)
        freq[i]=0;

    while(n=scanf("%d", &n))
    {
        if(n>0)
        {
            for(i=0; i<n; i++)
            {
                char s[100];
                scanf("%s", s);
                len=strlen(s);
                freq[len]++;
            }
        }
        else
            break;
    }


    printf("Lunghezza delle parole\n");
    for(i=1; i<MAXFREQ; i++)
        printf("%d: %d\n", i, freq[i]);
}

```

Istruzione while

Cicli while e annidati

Verifica "Media aritmetica"



media.c

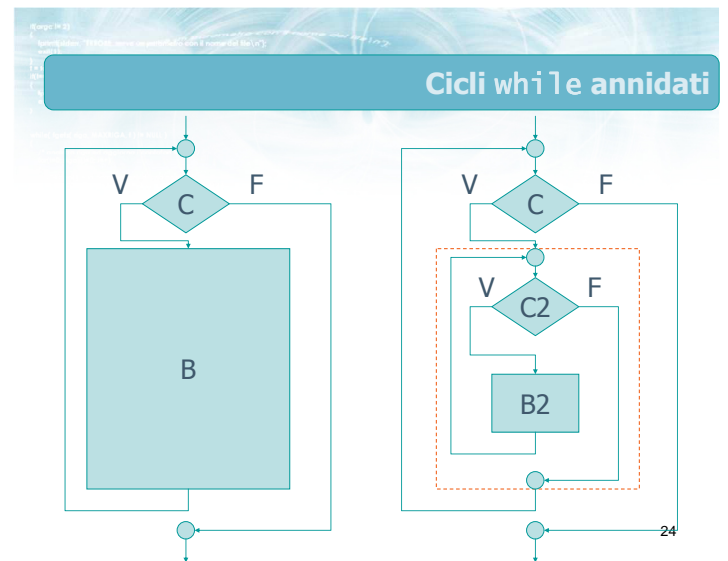
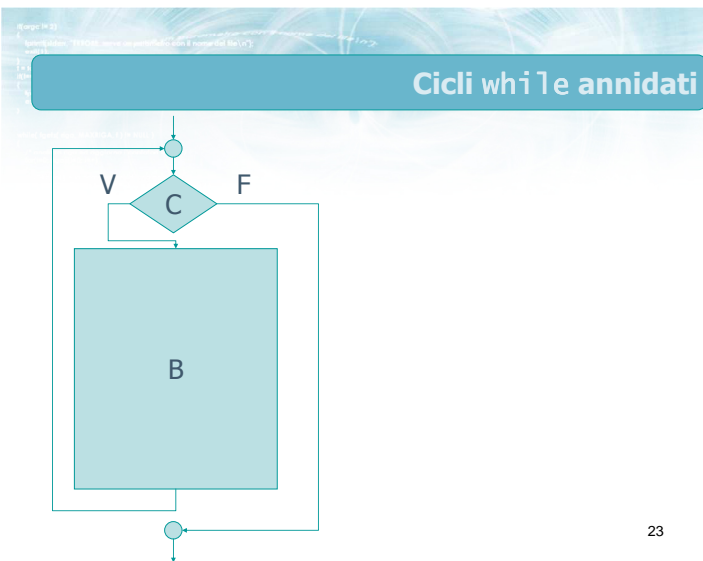
```

D:\Home\mirror\CDROM\materiale\W3C\media.exe
MEDIA ARITMETICA
Introduci n: 3
Ora inserisci 3 valori
Valore 1: 6.5
Valore 2: 3.2
Valore 3: 1.3
Risultato: 3.666667
Premere un tasto per continuare . . .

```

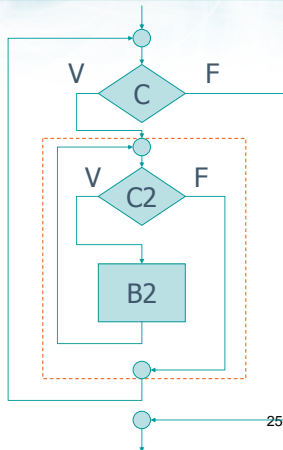
Annidamento di cicli

- All'interno del corpo del ciclo while è possibile racchiudere qualsiasi altra istruzione C
- In particolare, è possibile racchiudere un'istruzione while all'interno di un'altra istruzione while
- In tal caso, per ogni singola iterazione del ciclo while più esterno, vi saranno tutte le iterazioni previste per il ciclo più interno



Cicli while e annidati

```
while( C )
{
    while( C2 )
    {
        B2 ;
    }
}
```



25

Esempio

```
i = 0 ;
while( i < N )
{
    j = 0 ;
    while( j < N )
    {
        printf("i=%d - j=%d\n", i, j);
        j = j + 1 ;
    }
    i = i + 1 ;
}
```

conta99.c

26

Esempio

```
i = 0 ;
while( i < N )
{
    j = 0 ;
    while( j < N )
    {
        printf("i=%d - j=%d\n", i, j);
        j = j + 1 ;
    }
    i = i + 1 ;
}
```

conta99.c

```
i=0 - j=0
i=0 - j=1
i=0 - j=2
i=1 - j=0
i=1 - j=1
i=1 - j=2
i=2 - j=0
i=2 - j=1
i=2 - j=2
```

27

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA];
    char *parola;
    int i, j;

    for(i=0; i<MAXPAROLA; i++)
        freq[i] = 0;

    if(argc < 2)
        printf("Errore: serve specificare almeno 2 argomenti\n");
    else
    {
        parola = argv[1];
        int len = strlen(parola);
        for(i=0; i<len; i++)
            freq[(int)tolower(parola[i])]++;
    }

    printf("Parole e frequenze:\n");
    for(i=0; i<MAXPAROLA; i++)
        if(freq[i] > 0)
            printf("%s: %d\n", (char *)i, freq[i]);
}
```

Istruzione while

Esercizio "Quadrato"

Esercizio "Quadrato"

- Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Visualizzare un quadrato di lato n costituito da asterischi

29

Analisi

```
Pront di zmanidi
QUADRATO
Introduci n: 5
*****
*****
*****
*****
*****
```

30

Algoritmo

- Acquisisci n
- Ripeti n volte
 - Stampa una riga di n asterischi

31

Algoritmo

- Acquisisci n
- Ripeti n volte
 - Stampa una riga di n asterischi
 - Ripeti n volte
 - Stampa un singolo asterisco
 - Vai a capo

32

Traduzione in C

```
i = 0 ;
while( i < n )
{
    j = 0 ;
    while( j < n )
    {
        printf("*") ;
        j = j + 1 ;
    }
    printf("\n") ;
    i = i + 1 ;
}
```

quadrato.c

33

Traduzione in C

```
i = 0 ;
while( i < n )
{
    j = 0 ;
    while( j < n )
    {
        printf("*") ;
        j = j + 1 ;
    }
    printf("\n") ;
    i = i + 1 ;
}
```

quadrato.c

Ripeti n volte

Stampa una riga di n asterischi

34

Traduzione in C

```
i = 0 ;
while( i < n )
{
    j = 0 ;
    while( j < n )
    {
        printf("*") ;
        j = j + 1 ;
    }
    printf("\n") ;
    i = i + 1 ;
}
```

quadrato.c

Stampa n asterischi

Vai a capo

35

Traduzione in C

```
i = 0 ;
while( i < n )
{
    j = 0 ;
    while( j < n )
    {
        printf("*") ;
        j = j + 1 ;
    }
    printf("\n") ;
    i = i + 1 ;
}
```

quadrato.c

Ripeti n volte

Stampa un asterisco

36

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAXFREQ]; // vettore di contatori
    della frequenza delle lunghezze delle parole
    char parola[MAXPAROLA];
    int i, len, lunghezza;
    int f;

    for(i=0; i<MAXFREQ; i++)
        freq[i]=0;

    for(i=0; i<argc; i++)
    {
        if(strlen(argv[i]) > MAXPAROLA)
            continue;
        strcpy(parola, argv[i]);
        len = strlen(parola);
        f = freq[len];
        freq[len] = f+1;
    }

    printf("Parole: %d\n", freq[0]);
    return 0;
}
```

Cicli ed iterazioni

Schemi ricorrenti nei cicli

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAXFREQ]; // vettore di contatori
    della frequenza delle lunghezze delle parole
    char parola[MAXPAROLA];
    int i, len, lunghezza;
    int f;

    for(i=0; i<MAXFREQ; i++)
        freq[i]=0;

    for(i=0; i<argc; i++)
    {
        if(strlen(argv[i]) > MAXPAROLA)
            continue;
        strcpy(parola, argv[i]);
        len = strlen(parola);
        f = freq[len];
        freq[len] = f+1;
    }

    printf("Parole: %d\n", freq[0]);
    return 0;
}
```

Schemi ricorrenti nei cicli

Contatori

```
(argc < 2)
{
    printf("Errore: inserire almeno 2 numeri da 0 a 9\n");
    return 1;
}

for(i=0; i<argc; i++)
{
    int n = atoi(argv[i]);
    if(n < 0 || n > 9)
    {
        printf("Errore: inserire numeri da 0 a 9\n");
        return 1;
    }
    printf("Numero %d\n", n);
}
```

Schemi ricorrenti nei cicli

- Contatori
- Accumulatori
- Flag
- Esistenza e universalità

2

```
(argc < 2)
{
    printf("Errore: inserire almeno 2 numeri da 0 a 9\n");
    return 1;
}

for(i=0; i<argc; i++)
{
    int n = atoi(argv[i]);
    if(n < 0 || n > 9)
    {
        printf("Errore: inserire numeri da 0 a 9\n");
        return 1;
    }
    printf("Numero %d\n", n);
}
```

Contatori

- Spesso in un ciclo è utile sapere
 - Quante iterazioni sono state fatte
 - Quante iterazioni rimangono da fare
 - Quale numero di iterazione sia quella corrente
- Per questi scopi si usano delle "normali" variabili intere, dette **contatori**
 - Inizializzate prima del ciclo
 - Incrementate/decrementate ad ogni iterazione
 - Oppure incrementate/decrementate ogni volta che si riscontra una certa condizione

4

```
(argc < 2)
{
    printf("Errore: inserire almeno 2 numeri da 0 a 9\n");
    return 1;
}

for(i=0; i<argc; i++)
{
    int n = atoi(argv[i]);
    if(n < 0 || n > 9)
    {
        printf("Errore: inserire numeri da 0 a 9\n");
        return 1;
    }
    printf("Numero %d\n", n);
}
```

Contatori

```
int i, N ;
i = 0 ;
while ( i < N )
{
    printf("Iterazione %d\n", i+1) ;
    i = i + 1 ;
}
```

5

```
(argc < 2)
{
    printf("Errore: inserire almeno 2 numeri da 0 a 9\n");
    return 1;
}

for(i=0; i<argc; i++)
{
    int n = atoi(argv[i]);
    if(n < 0 || n > 9)
    {
        printf("Errore: inserire numeri da 0 a 9\n");
        return 1;
    }
    printf("Numero %d\n", n);
}
```

Esempio

- Scrivere un programma in C che
 - legga dall'utente 10 numeri interi
 - al termine dell'inserimento, stampi
 - quanti tra i numeri inseriti sono positivi
 - quanti tra i numeri inseriti sono negativi
 - quanti tra i numeri inseriti sono nulli

6

Soluzione (1/3)

```
int npos, nneg, nzero ;  
  
....  
npos = 0 ;  
nneg = 0 ;  
nzero = 0 ;
```

7

Soluzione (2/3)

```
i = 0 ;  
while( i < n )  
{  
    printf("Inserisci dato %d: ", i+1);  
    scanf("%d", &dato);  
  
    if( dato > 0 )  
        npos = npos + 1 ;  
    else if( dato < 0 )  
        nneg = nneg + 1 ;  
    else  
        nzero = nzero + 1 ;  
  
    i = i + 1 ;  
}
```

8

Soluzione (3/3)

```
printf("Numeri positivi: %d\n", npos);  
printf("Numeri negativi: %d\n", nneg);  
printf("Numeri nulli: %d\n", nzero);
```

9

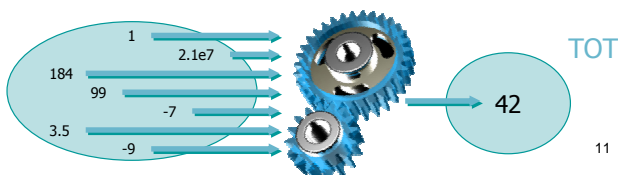


Schemi ricorrenti nei cicli

Accumulatori

Accumulatori (1/2)

- Spesso in un ciclo occorre calcolare un valore TOT che dipende dall'insieme dei valori analizzati nelle singole iterazioni
- Esempi:
 - TOT = sommatoria dei dati analizzati
 - TOT = produttoria dei dati analizzati
 - TOT = massimo, minimo dei dati analizzati



11

Accumulatori (2/2)

- In questo caso si usano delle variabili (interi o reali) dette **accumulatori**
 - Inizializzare TOT al valore che dovrebbe avere in assenza di dati (come se fosse n=0)
 - Ad ogni iterazione, aggiornare TOT tenendo conto del dato appena analizzato
 - Al termine del ciclo, TOT avrà il valore desiderato

12

Esempio: somma primi 10 interi

- Si scriva un programma in C che stampi il valore della somma dei primi 10 numeri interi

13

Analisi

- Inizializzazione di TOT
 - Qual è la somma dei primi 0 numeri interi?
 - $TOT = 0$
- Aggiornamento di TOT
 - Sapendo che TOT è la somma dei primi (i-1) numeri interi, e sapendo che il prossimo numero intero da sommare vale i, quanto dovrà valere TOT?
 - $TOT = TOT + i$

14

Soluzione: somma primi 10 interi

```
int tot ;
i = 1 ;
tot = 0 ;
while( i<=10 )
{
    tot = tot + i ;

    i = i + 1 ;
}

printf("La somma dei numeri da 1 a 10 ");
printf("vale %d\n", tot) ;
```

15

Esempio: fattoriale di K

- Si scriva un programma in C che, dato un numero intero K, calcoli e stampi il fattoriale di K
- $TOT = K!$

$$TOT = K! = \prod_{i=1}^{i=K} i$$

16

Analisi

- Inizializzazione di TOT
 - Qual è il valore del fattoriale per $K=0$?
 - $TOT = 1.0$
- Aggiornamento di TOT
 - Sapendo che TOT è pari al fattoriale di i-1, e sapendo che il prossimo numero da considerare è i, quanto dovrà valere TOT?
 - $TOT = TOT * i$

17

Soluzione: fattoriale di K

```
float tot ;
i = 1 ;
tot = 1.0 ;
while( i<=K )
{
    tot = tot * i ;

    i = i + 1 ;
}

printf("Il fattoriale di %d ", K);
printf("vale %f\n", tot) ;
```

18

Esempio: massimo

- Si scriva un programma in C che
 - acquisisca da tastiera N numeri reali
 - stampi il valore massimo tra i numeri acquisiti

19

Analisi

- Inizializzazione di TOT
 - Qual è il valore del massimo in un insieme di 0 numeri?
 - Non esiste, non è definito!
 - TOT = numero molto piccolo, che non possa certamente essere scambiato con il massimo
- Aggiornamento di TOT
 - Sapendo che TOT è pari al massimo dei primi $i-1$ dati, e sapendo che il prossimo dato da considerare è d , quanto dovrà valere TOT?
 - Se $d \leq \text{TOT}$, allora TOT rimane il massimo
 - Se $d > \text{TOT}$, allora il nuovo massimo sarà $\text{TOT} = d$

20

Esempio: massimo

```
int max ;  
  
i = 0 ;  
max = INT_MIN ;  
while( i < N )  
{  
    scanf("%d", &dato) ;  
    if(dato > max)  
        max = dato ;  
  
    i = i + 1 ;  
}  
  
printf("Massimo = %d\n", max);
```

21

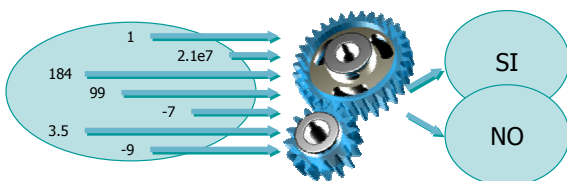


Schemi ricorrenti nei cicli

Flag

Flag, indicatori, variabili logiche

- Spesso occorre analizzare una serie di dati per determinare se si verifica una certa condizione
- Esempi:
 - Tra i dati inseriti esiste il numero 100?
 - Esistono due numeri consecutivi uguali?



23

Problemi

- Nel momento in cui si "scopre" il fatto, non si può interrompere l'elaborazione ma occorre comunque terminare il ciclo
- Al termine del ciclo, come fare a "ricordarsi" se si era "scoperto" il fatto o no?

24

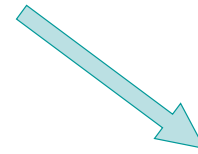
Una possibile soluzione

- Per sapere
 - Se una certa condizione si verifica è possibile contare
 - Quante volte quella condizione si verifica ed in seguito verificare
 - Verificare se il conteggio è diverso da zero
- Ci riconduciamo ad un problema risolvibile per mezzo di una variabile contatore

25

Esempio 1

Tra i dati inseriti esiste il numero 100?



Conta quante volte tra i dati inseriti compare il numero 100

Il conteggio è > 0 ?

26

Soluzione (1/3)

```
int i, n ;
int dato ;
int conta ;
/* conta il numero di "100" letti */
printf("TROVA 100\n");

n = 10 ;
printf("Inserisci %d numeri\n", n);
```

27

Soluzione (2/3)

```
conta = 0 ;
i = 0 ;
while( i < n )
{
    printf("Inserisci dato %d: ", i+1);
    scanf("%d", &dato);

    if( dato == 100 )
        conta = conta + 1 ;

    i = i + 1 ;
}
```

28

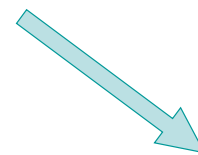
Soluzione (3/3)

```
if( conta != 0 )
    printf("Ho trovato il 100\n");
else
    printf("NON ho trovato il 100\n");
```

29

Esempio 2

Esistono due numeri consecutivi uguali?



Conta quante volte due numeri consecutivi sono uguali

Il conteggio è > 0 ?

30

Soluzione (1/3)

```
int i, n ;
int dato ;
int precedente ;
int conta ;
/* conta il numero di "doppioni" trovati */

printf("TROVA UGUALI\n");

n = 10 ;

printf("Inserisci %d interi\n", n);
```

31

Soluzione (2/3)

```
conta = 0 ;

precedente = INT_MAX ;
/* ipotesi: l'utente non lo inserira' mai */

i = 0 ;
while( i < n )
{
    printf("Inserisci dato %d: ", i+1);
    scanf("%d", &dato);

    if( dato == precedente )
        conta = conta + 1 ;

    precedente = dato ;

    i = i + 1 ;
}
```

32

Soluzione (3/3)

```
if( conta != 0 )
    printf("Ho trovato dei numeri
           consecutivi uguali\n");
else
    printf("NON ho trovato dei numeri
           consecutivi uguali\n");
```

33

Svantaggi

- Il contatore determina quante volte si verifica la condizione ricercata
- In realtà non mi serve sapere quante volte, ma solo se si è verificata almeno una volta
- Usiamo un contatore "degenere", che una volta arrivato ad 1 non si incrementa più

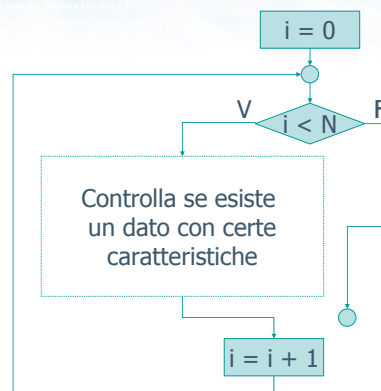
34

Variabili "flag"

- Variabili intere che possono assumere solo due valori
 - Variabile = 0 ⇒ la condizione non si è verificata
 - Variabile = 1 ⇒ la condizione si è verificata
- Viene inizializzata a 0 prima del ciclo
- Se la condizione si verifica all'interno del ciclo, viene posta a 1
- Al termine del ciclo si verifica il valore
- Sinonimi: Flag, Variabile logica, Variabile booleana, Indicatore

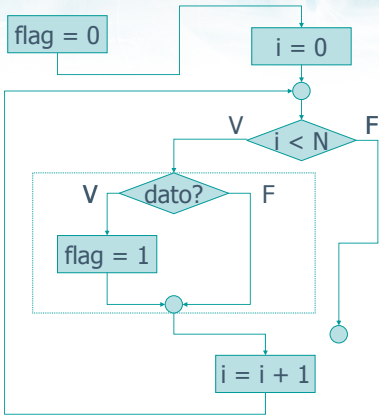
35

Analisi



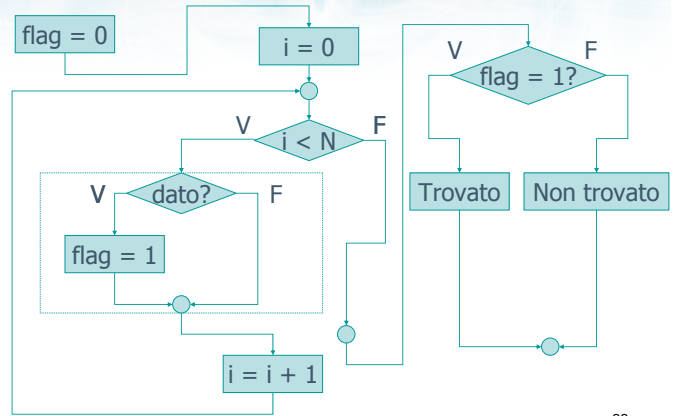
36

Analisi



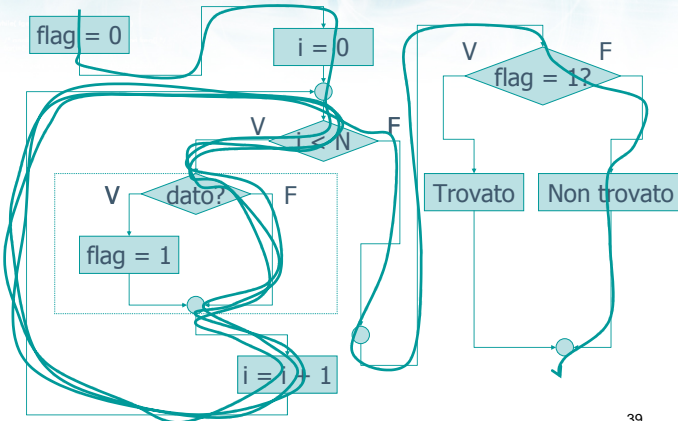
37

Analisi



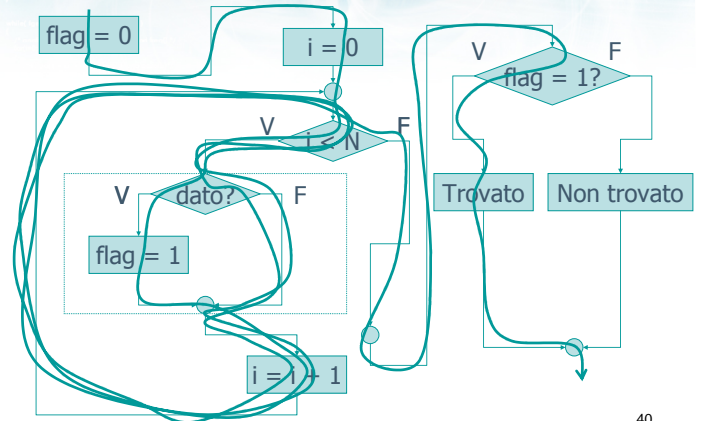
38

Analisi



39

Analisi



40

Soluzione con flag – esempio 1

```
int trovato ; /* ho visto il numero "100"? */
....
trovato = 0 ;

i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == 100 )
        trovato = 1 ;

    i = i + 1 ;
}

if( trovato != 0 )
    printf("Trovato il numero 100\n");
else
    printf("NON trovato il numero 100\n");
```

trova100v2.c

Soluzione con flag – esempio 2

```
int doppi ; /* trovati "doppioni" ? */
...
doppi = 0 ;

precedente = INT_MAX ;
i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == precedente )
        doppi = 1 ;

    precedente = dato ;
    i = i + 1 ;
}

if( doppi != 0 )
    printf("Trovati consecutivi uguali\n");
```

uguali2.c

Schemi ricorrenti nei cicli

Esistenza e universalità

Ricerca di esistenza o universalità

- L'utilizzo dei flag può essere utile quando si desiderino verificare delle proprietà su un certo insieme di dati
 - È vero che tutti i dati verificano la proprietà?
 - È vero che almeno un dato verifica la proprietà?
 - È vero che nessun dato verifica la proprietà?
 - È vero che almeno un dato non verifica la proprietà?

44

Esempi

- Verificare che tutti i dati inseriti dall'utente siano positivi
- Determinare se una sequenza di dati inseriti dall'utente è crescente
- Due numeri non sono primi tra loro se hanno almeno un divisore comune
 - esiste almeno un numero che sia divisore dei due numeri dati
- Un poligono regolare ha tutti i lati di lunghezza uguale
 - ogni coppia di lati consecutivi ha uguale lunghezza

45

Formalizzazione

- È vero che tutti i dati verificano la proprietà?
 - $\forall x : P(x)$
- È vero che almeno un dato verifica la proprietà?
 - $\exists x : P(x)$
- È vero che nessun dato verifica la proprietà?
 - $\forall x : \text{not } P(x)$
- È vero che almeno un dato non verifica la proprietà?
 - $\exists x : \text{not } P(x)$

46

Realizzazione (1/2)

- Esistenza: $\exists x : P(x)$
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata

47

Realizzazione (1/2)

- Esistenza: $\exists x : P(x)$
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata
- Universalità: $\forall x : P(x)$
 - Inizializzo flag $F = 1$
 - Ciclo su tutte le x
 - Se $P(x)$ è falsa
 - Pongo $F = 0$
 - Se $F = 1$, l'universalità è dimostrata
 - Se $F = 0$, l'universalità è negata

48

Realizzazione (2/2)

- Esistenza: $\exists x : \text{not } P(x)$
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è falsa
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata
- Universalità: $\forall x : \text{not } P(x)$
 - Inizializzo flag $F = 1$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 0$
 - Se $F = 1$, l'universalità è dimostrata
 - Se $F = 0$, l'universalità è negata

49

Esempio 1

- Verificare che tutti i dati inseriti dall'utente siano positivi

```
int positivi ;
...
positivi = 1 ;
i = 0 ;
while( i < n )
{
    ...
    if( dato <= 0 )
        positivi = 0 ;
    ...
    i = i + 1 ;
}
if( positivi == 1 )
    printf("Tutti positivi\n");
```

50

Esempio 2

- Determinare se una sequenza di dati inseriti dall'utente è crescente

```
int crescente ;
...
crescente = 1 ;
precedente = INT_MIN ;
i = 0 ;
while( i < n )
{
    ...
    if( dato < precedente )
        crescente = 0 ;
    precedente = dato ;
    ...
    i = i + 1 ;
}
```

51

Esempio 3

- Due numeri non sono primi tra loro se hanno almeno un divisore comune

```
int A, B ;
int noprimi ;
...
noprimi = 0 ;
i = 2 ;
while( i <= A )
{
    ...
    if( (A%i==0) && (B%i==0) )
        noprimi = 1 ;
    ...
    i = i + 1 ;
}
```

52

Esempio 4

- Un poligono regolare ha tutti i lati di lunghezza uguale

```
int rego ;
...
rego = 1 ;
precedente = INT_MIN ;
i = 0 ;
while( i < n )
{
    ...
    if( lato != precedente )
        rego = 0 ;
    precedente = lato ;
    ...
    i = i + 1 ;
}
```

53



Errore frequente

- Resetare il flag al valore di inizializzazione, dimenticando di fatto eventuali condizioni incontrate in precedenza

```
trovato = 0 ;
i = 0 ;
while( i < n )
{
    ...
    if( dato == 100 )
        trovato = 1 ;
    else
        trovato = 0 ;
    ...
    i = i + 1 ;
}

trovato = 0 ;
i = 0 ;
while( i < n )
{
    ...
    if( dato == 100 )
        trovato = 1 ;
    ...
    i = i + 1 ;
}
```



Errore frequente

- Passare ai fatti non appena trovato il primo elemento che soddisfa la proprietà

```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
    {  
        trovato = 1 ;  
        printf("W!\n");  
    }  
    ...  
    i = i + 1 ;  
}
```



```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
} if(trovato==1)  
    printf("W!\n");
```



Errore frequente

- Pensare che al primo fallimento si possa determinare che la proprietà è falsa

```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    else  
        printf("NO!\n");  
    ...  
    i = i + 1 ;  
}
```



```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
} if(trovato==0)  
    printf("NO!\n");
```



```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAXFREQ]; // vettore di contorni
    della frequenza delle lunghezze delle parole
    char dict[MAXDICT];
    int i, len, lunghezza;
    int r;
    for (i = 0; i < MAXFREQ; i++)
        freq[i] = 0;

    // legge le parole
    while (scanf("%s", dict) != EOF)
    {
        len = strlen(dict);
        if (len < 1 || len > MAXLEN)
            continue;
        r = freq[len];
        freq[len]++;
    }

    while (scanf("%s", dict) != EOF)
    {
        printf("%s\n", dict);
    }
}

```

Cicli ed iterazioni

Istruzione for

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[MAXFREQ]; // vettore di contorni
    della frequenza delle lunghezze delle parole
    char dict[MAXDICT];
    int i, len, lunghezza;
    int r;
    for (i = 0; i < MAXFREQ; i++)
        freq[i] = 0;

    // legge le parole
    while (scanf("%s", dict) != EOF)
    {
        len = strlen(dict);
        if (len < 1 || len > MAXLEN)
            continue;
        r = freq[len];
        freq[len]++;
    }

    while (scanf("%s", dict) != EOF)
    {
        printf("%s\n", dict);
    }
}

```

Istruzione for

Sintassi dell'istruzione

```

for ( I; C; A )
{
    B ;
}

```

5

```

for ( I; C; A )
{
    B ;
}

```

Istruzione for

- Sintassi dell'istruzione
- Operatori di autoincremento
- Cicli for annidati

2

```

for ( I; C; A )
{
    B ;
}

```

Istruzione for

- L'istruzione fondamentale è `while`
 - La condizione solitamente valuta una variabile di controllo
 - Occorre ricordarsi l'inizializzazione della variabile di controllo
 - Occorre ricordarsi di aggiornare (incrementare, ...) la variabile di controllo
- L'istruzione `for` rende più semplice ricordare queste cose

4

```

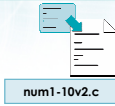
for ( I; C; A )
{
    B ;
}

```

6

Esempio

```
int i ;  
for ( i=1; i<=10; i=i+1 )  
{  
    printf("Numero = %d\n", i) ;  
}
```



7

Equivalenza for \leftrightarrow while

```
for ( I; C; A )  
{  
    B ;  
}
```

```
I ;  
while ( C )  
{  
    B ;  
    A ;  
}
```

8

Esempio

```
int i ;  
for ( i=1; i<=10; i=i+1 )  
{  
    printf("%d\n", i) ;  
}
```

```
int i ;  
i = 1 ;  
while ( i <= 10 )  
{  
    printf("%d\n", i) ;  
    i = i + 1 ;  
}
```

9

Utilizzo prevalente (1/2)

- ▶ Le istruzioni di inizializzazione **I** e di aggiornamento **A** possono essere qualsiasi
- ▶ Solitamente **I** viene utilizzata per inizializzare il contatore di controllo del ciclo, e quindi è del tipo
 - $i = 0$
- ▶ Solitamente **A** viene utilizzata per incrementare (o decrementare) il contatore, e quindi è del tipo
 - $i = i + 1$

```
for ( I; C; A )  
{  
    B ;  
}
```

10

Utilizzo prevalente (2/2)

- ▶ L'istruzione **for** può sostituire un qualsiasi ciclo **while**
- ▶ Solitamente viene utilizzata, per maggior chiarezza, nei cicli con numero di iterazioni noto a priori

11

Cicli for con iterazioni note

```
int i ;  
for ( i=0; i<N; i=i+1 )  
{  
    .....  
}
```

```
int i ;  
for ( i=1; i<=N; i=i+1 )  
{  
    .....  
}
```

```
int i ;  
for ( i=N; i>0; i=i-1 )  
{  
    .....  
}
```

```
int i ;  
for ( i=N-1; i>=0; i=i-1 )  
{  
    .....  
}
```

11

Casi particolari (1/6)

- Se non è necessario inizializzare nulla, si può omettere l'istruzione I
 - `for(; i != 0 ; i = i - 1)`
 - La condizione C viene comunque valutata prima della prima iterazione, pertanto le variabili coinvolte dovranno essere inizializzate prima dell'inizio del ciclo
 - Il simbolo ; è sempre necessario

```
for ( I; C; A )
{
    B ;
}
```

13

Casi particolari (2/6)

- Se l'aggiornamento viene fatto nel ciclo, si può omettere l'istruzione A
 - `for(dato = INT_MIN; dato != 0 ;)`
 - La responsabilità di aggiornare la variabile di controllo (dato) è quindi del corpo B del ciclo
 - Il simbolo ; è sempre necessario

```
for ( I; C; A )
{
    B ;
}
```

14

Casi particolari (3/6)

- Se occorre inizializzare più di una variabile, è possibile farlo separando le varie inizializzazioni con un simbolo ,
 - `for(i=0, j=0; i<N; i=i+1)`
 - Solitamente uno solo è il contatore del ciclo, gli altri saranno altri contatori, accumulatori o flag

```
for ( I; C; A )
{
    B ;
}
```

15

Casi particolari (4/6)

- Se occorre aggiornare più di una variabile, è possibile farlo separando i vari aggiornamenti con un simbolo ,
 - `for(i=0; i<N; i=i+1, k=k-1)`

```
for ( I; C; A )
{
    B ;
}
```

16

Casi particolari (5/6)

- Nel caso in cui si ometta sia I che A, il ciclo `for` degenera nel ciclo `while` e equivalente
 - `for(; i<N;)`
 - `while(i<N)`

```
for ( I; C; A )
{
    B ;
}
```

17

Casi particolari (6/6)

- È possibile omettere la condizione C, in tal caso viene considerata come sempre vera
 - `for(i=0; ; i=i+1)`
 - Questo costrutto genera un ciclo infinito. È necessario che il ciclo venga interrotto con un altro meccanismo (`break`, `return`, `exit`)
 - Talvolta si incontra anche un ciclo infinito "puro"
 - `for(; ;)`

```
for ( I; C; A )
{
    B ;
}
```

18

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; // vettore di contatori
    della frequenza della lunghezza delle parole
    char parola[MAXPAROLA];
    int i, nlen, lunghezza;
    FILE *f;

    printf("MAXPAROLA: %d\n", MAXPAROLA);
    printf("MAXRIGA: %d\n", MAXRIGA);

    // ...
}

```

Istruzione for

Operatori di autoincremento

Operatore di auto-incremento

`a++ ;`
`++a ;`

\Rightarrow

`a = a + 1 ;`

`a-- ;`
`--a ;`

\Rightarrow

`a = a - 1 ;`

21

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; // vettore di contatori
    della frequenza della lunghezza delle parole
    char parola[MAXPAROLA];
    int i, nlen, lunghezza;
    FILE *f;

    printf("MAXPAROLA: %d\n", MAXPAROLA);
    printf("MAXRIGA: %d\n", MAXRIGA);

    // ...
}

```

Istruzione for

Cicli for annidati

Istruzione di aggiornamento

- Nella maggioranza dei casi, l'istruzione di aggiornamento **A** consiste in un incremento
 - $i = i + 1$
- oppure in un decremento
 - $i = i - 1$
- Il linguaggio **C** dispone di operatori specifici per semplificare la sintassi di queste operazioni frequenti

```

for ( I; C; A )
{
    B ;
}

```

20

Cicli for con iterazioni note

```

int i ;
for ( i=0; i<N; i++ )
{
    .....
}

```

```

int i ;
for ( i=1; i<=N; i++ )
{
    .....
}

```

```

int i ;
for ( i=N; i>0; i-- )
{
    .....
}

```

```

int i ;
for ( i=N-1; i>=0; i-- )
{
    .....
}

```

Annidamento di cicli

- Come sempre, all'interno del corpo **B** di un ciclo (`for` o `while`) è possibile annidare altri cicli (`for` o `while`)
- Non vi è limite al livello di annidamento
- I cicli più interni sono sempre eseguiti "più velocemente" dei cicli più esterni

24

Esempio

```
for( i=0; i<N; i++ )
{
    for( j=0; j<N; j++ )
    {
        printf("i=%d - j=%d\n", i, j);
    }
}
```

25

Esercizio

- Si scriva un programma in linguaggio C che
 - acquisisca da tastiera 10 numeri
 - per ciascuno di tali numeri determini se è un numero primo, stampando immediatamente un messaggio opportuno
 - al termine, se nessuno tra i numeri inseriti era un numero primo, stampi un messaggio opportuno

26

Analisi (1/2)

```
Prompt dei comandi

TROVA PRIMI
Inserisci 4 numeri interi

Inserisci dato 1: 6
Inserisci dato 2: 3
E' un numero primo
Inserisci dato 3: 4
Inserisci dato 4: 5
E' un numero primo
```

27

Analisi (2/2)

```
Prompt dei comandi

TROVA PRIMI
Inserisci 4 numeri interi

Inserisci dato 1: 4
Inserisci dato 2: 6
Inserisci dato 3: 8
Inserisci dato 4: 9

Non c'erano numeri primi
```

28

Numero primo

```
primo = 1 ;
for( j=2; j<dato; j++)
{
    if( dato%j == 0 )
        primo = 0 ;
}
```

29

Stampa se non ci sono primi

```
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    ....determina se è un numero primo....

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
    }
}
```

30

Stampa se è un primo

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    ....acquisisci dato e determina se è un numero primo....
    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non ci sono primi\n") ;
```

31

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non c'erano numeri primi\n");
```

32

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non c'erano numeri primi\n");
```

Ciclo esterno

Ciclo interno

33

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non c'erano numeri primi\n");
```

Flag interno:
numero primo

Inizializzazione

Aggiornamento

Verifica

34

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
printf("Non c'erano numeri primi\n");
```

Flag esterno:
nessun primo

Inizializzazione

Aggiornamento

Verifica

35

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; // vettore di contatore
    // della frequenza delle lunghezze delle parole
    char parola[MAXRIGA];
    int i, len, lunghezza;
    int r;

    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;

    while (r=0)
    {
        printf("Inserisci una parola (o 'q' per uscire): ");
        fgets(parola, MAXRIGA, stdin);
        len = strlen(parola);
        if (parola[len-1] != '\n')
            len--;
        if (len > 0)
        {
            freq[len]++;
        }
        if (parola[0] != 'q')
            printf("Parola: %s\n", parola);
    }

    printf("Frequenza delle lunghezze delle parole:\n");
    for(i=1; i<MAXPAROLA; i++)
        printf("%d: %d\n", i, freq[i]);

    return 0;
}

```

Cicli ed iterazioni

Approfondimenti

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; // vettore di contatore
    // della frequenza delle lunghezze delle parole
    char parola[MAXRIGA];
    int i, len, lunghezza;
    int r;

    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;

    while (r=0)
    {
        printf("Inserisci una parola (o 'q' per uscire): ");
        fgets(parola, MAXRIGA, stdin);
        len = strlen(parola);
        if (parola[len-1] != '\n')
            len--;
        if (len > 0)
        {
            freq[len]++;
        }
        if (parola[0] != 'q')
            printf("Parola: %s\n", parola);
    }

    printf("Frequenza delle lunghezze delle parole:\n");
    for(i=1; i<MAXPAROLA; i++)
        printf("%d: %d\n", i, freq[i]);

    return 0;
}

```

Approfondimenti

Istruzione do-while

```

// ...
while (condizione)
{
    // ...
}
// ...

```

Approfondimenti

- Istruzione do-while
- Istruzione break
- Istruzione continue

2

```

// ...
do
{
    // ...
} while (condizione);
// ...

```

Istruzione do-while

```

do {
    B ;
} while ( C ) ;

```

4

Confronto

<ul style="list-style-type: none"> ➤ Istruzione while <ul style="list-style-type: none"> ● Condizione valutata prima di ogni iterazione ● Numero minimo di iterazioni: 0 ● Per uscire: condizione falsa 	<ul style="list-style-type: none"> ➤ Istruzione do-while <ul style="list-style-type: none"> ● Condizione valutata al termine di ogni iterazione ● Numero minimo di iterazioni: 1 ● Per uscire: condizione falsa
--	--

5

Esempio

- Acquisire un numero compreso tra 1 e 10 da tastiera
- Nel caso in cui l'utente non inserisca il numero correttamente, chiederlo nuovamente

6

Soluzione

```
printf("Numero tra 1 e 10\n");
do {
    scanf("%d", &n) ;
} while ( n<1 || n>10 ) ;
```

7

Soluzione migliore

```
printf("Numero tra 1 e 10\n");
do {
    scanf("%d", &n) ;
    if( n<1 || n>10 )
        printf("Errore: ripeti\n");
} while ( n<1 || n>10 ) ;
```

8

Esempio

- ▶ Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- ▶ La sequenza termina quando l'utente inserisce il dato 9999

9

Soluzione

```
somma = 0 ;
do {
    scanf("%d", &dato) ;
    if( n != 9999 )
        somma = somma + dato ;
} while ( dato != 9999 ) ;
```

10

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXCICLO 50

int main(int argc, char *argv[])
{
    int i;
    char parola[MAXPAROLA];
    int ciclo;
    int n;
    int summa;

    printf("Inserisci una parola e un numero da 1 a 50\n");
    scanf("%s %d", parola, &ciclo);
    while (ciclo > 0)
    {
        printf("Inserisci un numero da 1 a 10\n");
        scanf("%d", &n);
        summa += n;
        ciclo--;
    }
    printf("La somma dei numeri da 1 a %d è %d\n", ciclo, summa);
    return 0;
}
```

Approfondimenti

Interruzione dei cicli

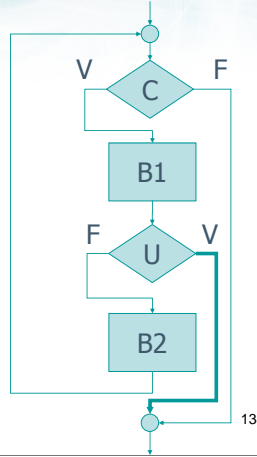
- ▶ Di norma, un ciclo termina quando la condizione di controllo diventa falsa
 - Necessario arrivare al termine del corpo per poter valutare la condizione
- ▶ Talvolta potrebbe essere comodo interrompere prematuramente l'esecuzione di un ciclo
 - A seguito di condizioni di errore
 - Quando è stato trovato ciò che si cercava

Istruzione break

12

Istruzione break

```
while ( C )
{
    B1 ;
    if ( U )
        break ;
    B2 ;
}
```



Funzionamento

- ▶ Quando viene eseguita l'istruzione break
 - Viene interrotta l'esecuzione del corpo del ciclo
 - Il flusso di esecuzione passa all'esterno del ciclo che contiene il break
 - Si esce dal ciclo anche se la condizione di controllo è ancora vera
 - In caso di cicli annidati, si esce solo dal ciclo più interno
- ▶ Funziona con cicli while, for, do-while

14

Esempio

- ▶ Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- ▶ La sequenza termina quando l'utente inserisce il dato 9999

15

Soluzione

```
somma = 0 ;
do {
    scanf("%d", &dato) ;

    if( dato == 9999 )
        break;

    somma = somma + dato ;
} while ( 1 ) ;
```

16

Esempio

- ▶ Si scriva un programma in C che determini se un numero inserito da tastiera è primo

17

Soluzione

```
scanf("%d", &dato) ;
primo = 1 ;
for ( i=2; i<dato; i++ )
{
    if( dato%i == 0 )
    {
        primo = 0 ;
        break ; /* inutile continuare */
    }
}
```

18


```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[256]; // vettore di conteggio
    della frequenza delle lunghezze delle parole
    char s[1024];
    int i, len, lunghezza;
    int r;

    scanf("%s", s);

    for (i = 0; i < 256; i++)
        freq[i] = 0;

    for (i = 0; i < strlen(s); i++)
        freq[(int)s[i]]++;

    printf("Frequenza massima: %d\n", *max_element(freq, freq + 256));

    return 0;
}

void main() {
    scanf("%s", s);
}
```

Cicli ed iterazioni

Esercizi proposti

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char *argv[])
{
    int freq[256]; // vettore di conteggio
    della frequenza delle lunghezze delle parole
    char s[1024];
    int i, len, lunghezza;
    int r;

    scanf("%s", s);

    for (i = 0; i < 256; i++)
        freq[i] = 0;

    for (i = 0; i < strlen(s); i++)
        freq[(int)s[i]]++;

    printf("Frequenza massima: %d\n", *max_element(freq, freq + 256));

    return 0;
}

void main() {
    scanf("%s", s);
}
```

Esercizi proposti

Esercizio "Decimale-binario"

Analisi

```
DECIMALE - BINARIO
Inserisci un numero intero positivo: 12
Numero binario: 0 0 1 1
```

Esercizi proposti

- Esercizio "Decimale-binario"
- Esercizio "Massimo Comun Divisore"
- Esercizio "Triangolo di Floyd"

2

Esercizio "Decimale-binario"

- Si realizzi un programma in C in grado di
 - Leggere un numero naturale n
 - Convertire tale numero dalla base 10 alla base 2
 - Visualizzare il risultato, a partire dalla cifra meno significativa

4

Divisioni successive

- $n = 12$
- $n \% 2 = 0 \rightarrow$ cifra 0
- $n = n / 2 = 6$
- $n \% 2 = 0 \rightarrow$ cifra 0
- $n = n / 2 = 3$
- $n \% 2 = 1 \rightarrow$ cifra 1
- $n = n / 2 = 1$
- $n \% 2 = 1 \rightarrow$ cifra 1
- $n = n / 2 = 0 \rightarrow$ STOP

N	N % 2
12	0
6	0
3	1
1	1
0	

6

Soluzione

```
while( n!=0 )
{
    if( n%2 == 1 )
        printf("1 ") ;
    else
        printf("0 ") ;

    n = n / 2 ;
}
```

bin-dec.c

7

Esercizi proposti

Esercizio "Massimo Comun Divisore"

Esercizio "Massimo Comun Divisore"

- Si scriva un programma in C in grado di calcolare il massimo comun divisore (MCD) di due numeri interi.
- Il MCD è definito come il massimo tra i divisori comuni ai due numeri.

9

Analisi

- Diciamo N1 e N2 i numeri inseriti dall'utente
- Il MCD di N1 e N2 è il **massimo** tra i numeri che sono **divisori** sia di N2, sia di N1.
 - Troviamo i divisori di N1 ...
 - ... tra quelli che sono anche divisori di N2 ...
 - ... calcoliamo il massimo

10

Algoritmo

- $k_max = 0$
- for $k =$ da 1 a $N1$
 - se k è un divisore di $N1$
 - se k è un divisore di $N2$
 - aggiorna $k_max = k$
- $MCD = k_max$

bin-dec.c

11

Esercizi proposti

Esercizio "Triangolo di Floyd"

Esercizio "Triangolo di Floyd"

- Scrivere un programma C per la rappresentazione del triangolo di Floyd.
- Il programma riceve da tastiera un numero interno N.
- Il programma visualizza le prima N righe del triangolo di Floyd.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

N=5

13

Analisi

- Occorre stampare i primi numeri interi in forma di triangolo
- La riga k-esima ha k elementi

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

N=5

14

Algoritmo

- cont = 1
- for riga = da 1 a N
 - for colonna = da 1 a riga
 - stampa cont
 - cont++
 - vai a capo



floyd.c

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

N=5

15

Cicli ed iterazioni

Sommario

Argomenti trattati

- Ripetizione del del flusso di esecuzione
- Inizializzazione, Condizione, Aggiornamento, Corpo
- Istruzione `while`
- Istruzione `for`
- Cicli annidati

2

Tecniche di programmazione

- Cicli con numero di iterazioni note o ignote
- Contatori
- Accumulatori
- Flag

3

Schemi ricorrenti

- Calcolo di somme, medie, ...
- Calcolo di max, min
- Ricerca di esistenza
- Ricerca di universalità
- Controllo dei dati in input

4

Suggerimenti

- Ricordare di verificare sempre le 4 parti del ciclo
 - Inizializzazione, Condizione, Corpo, Aggiornamento
- Le complicazioni nascono da
 - Cicli annidati
 - Condizioni `if` annidate in cicli
 - Annidamento di flag o ricerche
- Procedere sempre per gradi
 - Pseudo-codice o flow chart
 - Identificare chiaramente il ruolo dei diversi cicli

5

Materiale aggiuntivo

- Sul CD-ROM
 - Testi e soluzioni degli esercizi trattati nei lucidi
 - Scheda sintetica
 - Esercizi risolti
 - Esercizi proposti
- Esercizi proposti da altri libri di testo

6