

Introduzione al linguaggio C

Laboratorio di
Linguaggi di Programmazione
a.a. 2001/2002

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Prerequisiti ed obiettivi

- Programmazione strutturata
- Teorema di Bohm-Jacopini
- Esperienza di programmazione in Pascal

- Operatività in tempi ridotti
- Approfondimenti *in itinere*

Testi consigliati

- “Linguaggi di programmazione”
di R. Sethi
edito da Zanichelli
- “Linguaggio C (ANSI C)” – II ed.
di B.W. Kernighan, D.M. Ritchie
edito da Jackson Libri
- “C - Corso completo di programmazione”
di Harvey Deitel, Paul Deitel
edito da Apogeo

Sommario (I parte)

- Richiami alla Programmazione Strutturata
- Caratteristiche di base del C
- Tipi primitivi
- Assegnamento
- Operatori aritmetici
- Operatori logici e relazionali
- Strutture di controllo
 - Sequenza
 - Selezione (if, if/else, switch)
 - Iterazione (while, do/while, for)

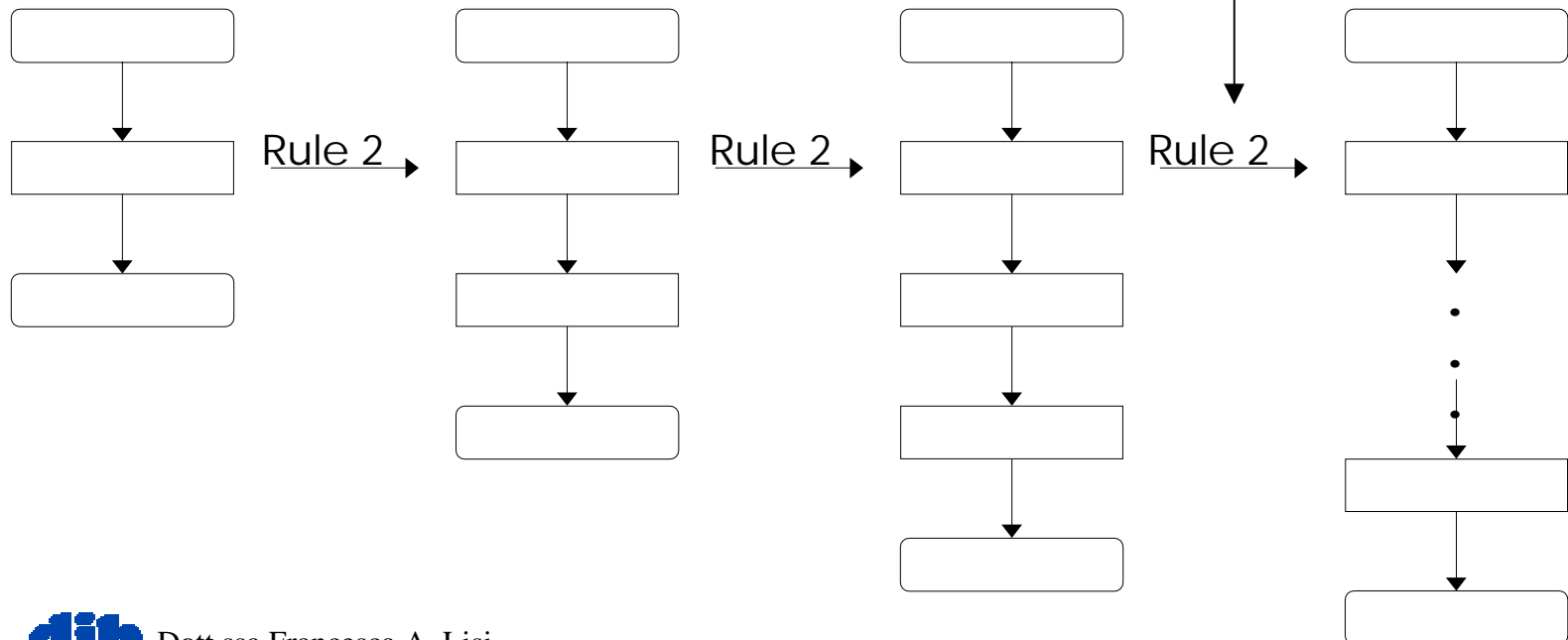
Richiami alla Programmazione Strutturata

- Rende i programmi più facili da comprendere, testare, correggere e modificare
- Utilizza solo strutture di controllo singolo-ingresso /singola-uscita
- Si basa su alcune **regole** pratiche:
 - 1) Comincia con il tracciare il “flowchart” più semplice
 - 2) Qualsiasi azione (blocco rettangolare) può essere rimpiazzato da due azioni in sequenza.
 - 3) Qualsiasi azione (blocco rettangolare) può essere rimpiazzato da una qualunque struttura di controllo (sequenza, selezione, iterazione).
 - 4) Le regole 2 e 3 possono essere applicate in qualsiasi ordine e in un numero svariato di volte.

Richiami alla Programmazione Strutturata (II)

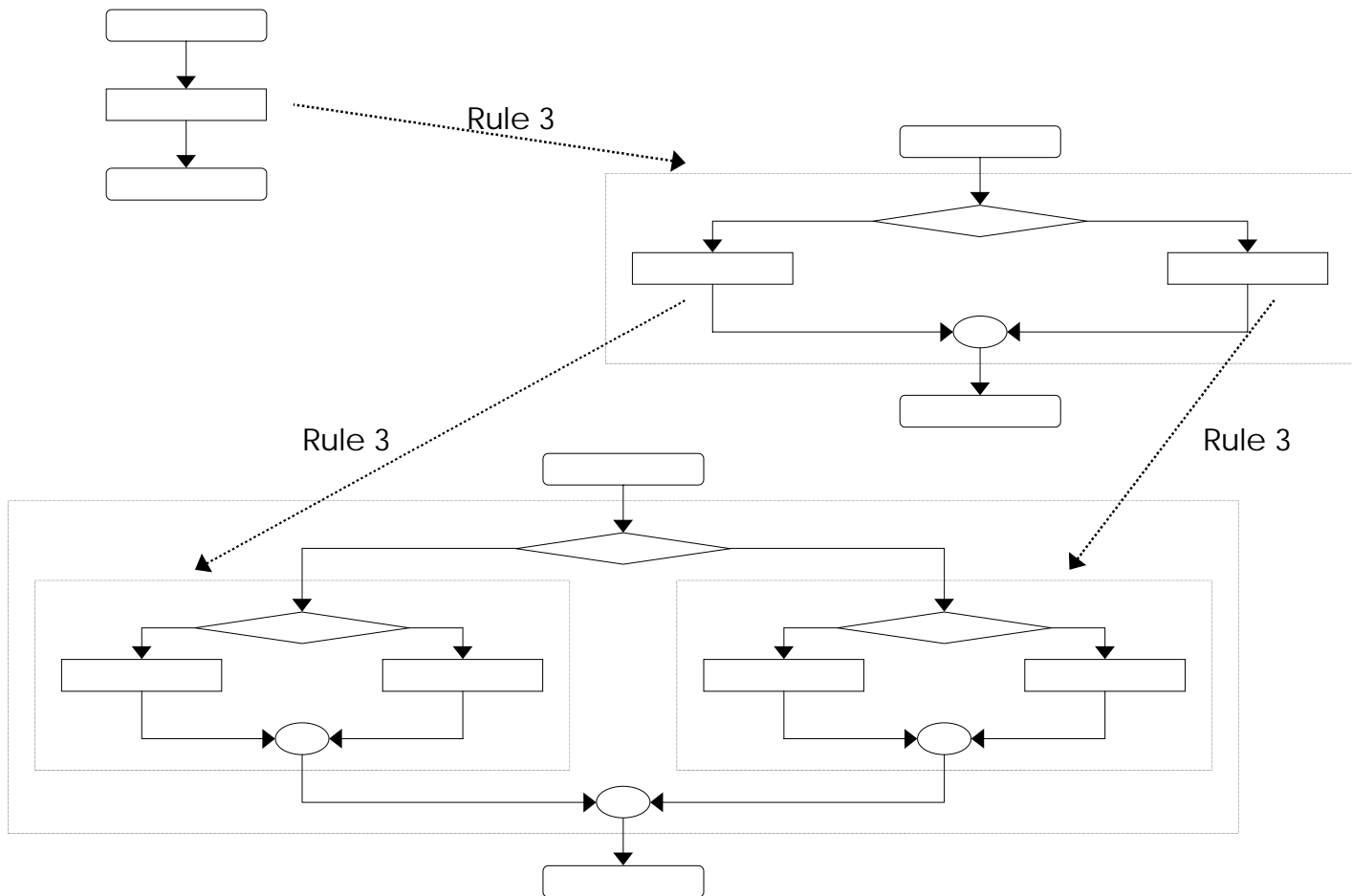
Rule 1 - Begin with the simplest flowchart

Rule 2 - Any rectangle can be replaced by two rectangles in sequence



Richiami alla Programmazione Strutturata (III)

Rule 3 - Replace any rectangle with a control structure



Caratteristiche di base del C

- Tendenza al basso livello
 - Corrispondenza con le istruzioni macchina
 - Efficienza
 - Sviluppato originariamente per UNIX, ora utilizzato per scrivere codice dei sistemi operativi
- Standard ANSI (1989-1999)
 - Definizione non ambigua
 - Indipendenza dalla macchina (portabilità)
- Tipizzazione debole
 - Controlli laschi
 - Comportamenti imprevisti

Un semplice programma C

```
1  /* Fig. 2.1: fig02_01.c
2     A first program in C */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "Welcome to C!\n" );
8
9     return 0;
10 }
```

Welcome to C!

Un semplice programma C (II)

- Testo delimitato da `/*` e `*/`
 - Racchiude commenti
 - Viene ignorato dal compilatore
- `#include <stdio.h>`
 - `#include` è una direttiva al pre-processor
 - `<stdio.h>` è un file di libreria che consente le operazioni I/O standard
 - Carica i contenuti del file di libreria

Un semplice programma C (III)

- **int main()**

- I programmi C contengono una o più funzioni, di cui una esattamente deve essere **main()**
- Le parentesi tonde () stanno ad indicare la parametricità della funzione
- **int** significa che **main()** restituisce un valore intero
- Le parentesi graffe { } racchiudono il corpo della funzione

- **printf("Welcome to C! \n");**

- Chiede al calcolatore di stampare una stringa di caratteri fra virgolette e di andare a capo (**\n**)
- L'intera riga di codice è detta "statement"
 - Tutti gli "statement" devono terminare con un punto e virgola

Tipi primitivi

Tipo	Tipo predefinito Pascal	Tipo predefinito C
numeri interi	<code>integer</code>	<code>int</code>
numeri reali a precisione singola	<code>real</code>	<code>float</code>
numeri reali a precisione doppia		<code>double</code>
caratteri	<code>char</code>	<code>char</code>
booleano	<code>Boolean</code>	<code>int (0=false)</code>

- Modificatori di tipo
 - `signed` e `unsigned` (si applicano a `int` e `char`)
 - `short` (si applica a `int`)
 - `long` (si applica a `int` e `double`)

Operatore di assegnamento =

Cfr Pascal :=

- **l-valori** (lvalues)
 - Espressioni che possono comparire sulla sinistra di un'equazione
 - I loro valori possono essere cambiati (es. $\mathbf{x=4}$ lecito)
- **r-valori** (rvalues)
 - Espressioni che possono solo apparire sulla destra di un'equazione
 - I loro valori sono costanti (es. $\mathbf{4=x}$ illecito)
- Gli l-valori possono essere usati come r-valori, ma non viceversa ($\mathbf{y=x}$)

Operatori aritmetici

Operazione	Operatore Pascal	Operatore C	Tipo operandi	Tipo risultato
somma	+	+	qualsiasi tipo aritmetico	stesso tipo dell'operando con tipo "superiore"
differenza	-	-		
prodotto	*	*		
divisione	/	/		
divisione fra interi	div		intero	intero
modulo	mod	%		

- Regole di precedenza

- ()
- *, /, % (se ce ne sono diversi, vanno valutati da sinistra verso destra)
- +, - (se ce ne sono diversi, vanno valutati da sinistra verso destra)

- Regole di conversione

- long double
- double
- float
- unsigned long int
- long int
- unsigned int
- int



Outline



1. Initialize variables

2. Input

2.1 Sum

3. Print

```
1  /* Fig. 2.5: fig02_05.c
2     Addition program */
3  #include <stdio.h>
4
5  int main()
6  {
7     int integer1, integer2, sum;      /* declaration */
8
9     printf( "Enter first integer\n" ); /* prompt */
10    scanf( "%d", &integer1 );         /* read an integer */
11    printf( "Enter second integer\n" ); /* prompt */
12    scanf( "%d", &integer2 );         /* read an integer */
13    sum = integer1 + integer2;        /* assignment of sum */
14    printf( "Sum is %d\n", sum );     /* print sum */
15
16    return 0; /* indicate that program ended successfully */
17 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Program Output

Un altro semplice programma C

- Come nel precedente esempio
 - Commenti, `#include <stdio.h>` e `main`
- `int integer1, integer2, sum;`
 - Dichiarazione di variabili (devono apparire prima delle istruzioni eseguibili)
 - `int` significa che le variabili possono assumere valori di tipo intero
 - `integer1`, `integer2`, e `sum` sono identificatori di variabili

Un altro semplice programma C (II)

- `scanf("%d", &integer1);`
 - Ottiene un valore dall'utente attraverso il canale di input standard (di solito, la tastiera)
 - Questa invocazione di **scanf** ha due argomenti:
 - **%d** - indica che i dati dovrebbero essere un intero decimale
 - **&integer1** - indirizzo di una locazione in memoria (variabile) denominata **integer1**
 - **&** va obbligatoriamente usato nelle chiamate di **scanf** (se ne parlerà più in là)
 - L'utente risponde a **scanf** digitando il numero, quindi battendo invio

Un altro semplice programma C (III)

- **sum = integer1 + integer2;**
 - **sum** è un l-valore
 - **integer1 + integer2** è un r-valore
 - Assegnamento del valore risultato dell'espressione **integer1 + integer2** alla variabile **sum**
- **printf("Sum is %d\n", sum);**
 - Simile a **scanf**
 - **%d** significa che verrà stampato un intero decimale (in questo caso, il valore della variabile **sum**)

Un altro semplice programma C (IV)

- **return 0;**
 - Uno dei modi per uscire da una funzione
 - **0** significa che il programma è terminato normalmente
- **}**
 - Indica che la fine di **main** è stata raggiunta

Operatori logici e relazionali

Operazione	Operatore Pascal	Operatore C	Tipo operandi	Tipo risultato
prodotto logico	and	&&	booleano	booleano
somma logica	or	 	booleano	booleano
negazione logica	not	!	booleano	booleano

Operazione	Operatore Pascal	Operatore C	Tipo operandi	Tipo risultato
uguaglianza	=	==	qualsiasi tipo	booleano
disuguaglianza	<>	!=	semplice	
minoranza stretta	<	<	tipo semplice o	
maggioranza stretta	>	>	stringhe	
minoranza	<=	<=	tipo semplice	
maggioranza	>=	>=		

Lo zucchero sintattico del C

- Operatori di incremento e decremento $++$ e $--$
 - $var ++$ equivale a $var = var + 1$
 - $var --$ equivale a $var = var - 1$
 - $++ var, -- var$
- Operatori di assegnamento $op=$
 $espr-1 op= espr-2$
equivale a
 $espr-1 = (espr-1) op (espr-2)$
 - $+, -, /, \%, \ll, \gg, \&, ^, |$
 - es. **`i += 2`**

La struttura di sequenza

Pascal

C

begin

istruzione-1;

istruzione-2;

...;

istruzione-n

end

{

istruzione-1;

istruzione-2;

...;

istruzione-n;

}

Le strutture di selezione `if` e `if/else`

Pascal

```
if espressione  
  then istruzione
```

```
if espressione  
  then istruzione1  
  else istruzione2
```

C

```
if (espressione)  
  istruzione
```

```
if (espressione)  
  istruzione1  
  else istruzione2
```



Outline



1. Declare variables

2. Input

2.1 if statements

3. Print

```
1  /* Fig. 2.13: fig02_13.c
2     Using if statements, relational
3     operators, and equality operators */
4  #include <stdio.h>
5
6  int main()
7  {
8     int num1, num2;
9
10    printf( "Enter two integers, and I will tell you\n" );
11    printf( "the relationships they satisfy: " );
12    scanf( "%d%d", &num1, &num2 );    /* read two integers */
13
14    if ( num1 == num2 )
15        printf( "%d is equal to %d\n", num1, num2 );
16
17    if ( num1 != num2 )
18        printf( "%d is not equal to %d\n", num1, num2 );
19
20    if ( num1 < num2 )
21        printf( "%d is less than %d\n", num1, num2 );
22
23    if ( num1 > num2 )
24        printf( "%d is greater than %d\n", num1, num2 );
25
26    if ( num1 <= num2 )
27        printf( "%d is less than or equal to %d\n",
28            num1, num2 );
```




Outline



3.1 Exit main

```
29
30     if ( num1 >= num2 )
31         printf( "%d is greater than or equal to %d\n",
32                 num1, num2 );
33
34     return 0;    /* indicate program ended successfully */
35 }
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

Program Output

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

La struttura di selezione multipla `switch`

Pascal

```
case espressione of  
  cost-1 : istruzione  
  cost-2 : istruzione  
  ...  
  cost-n : istruzione  
end
```

C

```
switch (espressione)  
{  
  case cost-1 : istruzione  
  case cost-2 : istruzione  
  ...  
  case cost-n : istruzione  
  default : istruzione  
}
```

Le strutture di iterazione `while` e `do/while`

Pascal

```
while espressione do  
    istruzione ;
```

repeat

istruzione

```
until (espressione) ;
```

C

```
while (espressione)  
    istruzione
```

do

istruzione

```
while (espressione)
```



Outline



1. Initialize Variables
2. Execute Loop
3. Output results

```
1  /* Fig. 3.6: fig03_06.c
2     Class average program with
3     counter-controlled repetition */
4  #include <stdio.h>
5
6  int main()
7  {
8     int counter, grade, total, average;
9
10    /* initialization phase */
11    total = 0;
12    counter = 1;
13
14    /* processing phase */
15    while ( counter <= 10 ) {
16        printf( "Enter grade: " );
17        scanf( "%d", &grade );
18        total = total + grade;
19        counter = counter + 1;
20    }
21
22    /* termination phase */
23    average = total / 10;
24    printf( "Class average is %d\n", average );
25
26    return 0;    /* indicate program ended successfully */
27 }
```



Outline



Program Output

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```



Outline



1. Initialize variables
2. Input data and count passes/failures
3. Print results

```
1  /* Fig. 3.10: fig03_10.c
2     Analysis of examination results */
3  #include <stdio.h>
4
5  int main()
6  {
7     /* initializing variables in declarations */
8     int passes = 0, failures = 0, student = 1, result;
9
10    /* process 10 students; counter-controlled loop */
11    while ( student <= 10 ) {
12        printf( "Enter result ( 1=pass,2=fail ): " );
13        scanf( "%d", &result );
14
15        if ( result == 1 )          /* if/else nested in while */
16            passes = passes + 1;
17        else
18            failures = failures + 1;
19
20        student = student + 1;
21    }
22
23    printf( "Passed %d\n", passes );
24    printf( "Failed %d\n", failures );
25
26    if ( passes > 8 )
27        printf( "Raise tuition\n" );
28
29    return 0;    /* successful termination */
30 }
```



Outline



Program Output

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```



1. Initialize variables

2. Input data

2.1 Use switch loop to update count

```
1  /* Fig. 4.7: fig04_07.c
2     Counting letter grades */
3  #include <stdio.h>
4
5  int main()
6  {
7     int grade;
8     int aCount = 0, bCount = 0, cCount = 0,
9         dCount = 0, fCount = 0;
10
11    printf( "Enter the letter grades.\n" );
12    printf( "Enter the EOF character to end input.\n" );
13
14    while ( ( grade = getchar() ) != EOF ) {
15
16        switch ( grade ) {      /* switch nested in while */
17
18            case 'A': case 'a': /* grade was uppercase A */
19                ++aCount;      /* or lowercase a */
20                break;
21
22            case 'B': case 'b': /* grade was uppercase B */
23                ++bCount;      /* or lowercase b */
24                break;
25
26            case 'C': case 'c': /* grade was uppercase C */
27                ++cCount;      /* or lowercase c */
28                break;
29
30            case 'D': case 'd': /* grade was uppercase D */
31                ++dCount;      /* or lowercase d */
32                break;
```




2.1 Use switch loop to update count

3. Print results

```
33
34     case 'F': case 'f': /* grade was uppercase F */
35         ++fCount;      /* or lowercase f */
36         break;
37
38     case '\n': case ' ': /* ignore these in input */
39         break;
40
41     default: /* catch all other characters */
42         printf( "Incorrect letter grade entered." );
43         printf( " Enter a new grade.\n" );
44         break;
45     }
46 }
47
48 printf( "\nTotals for each letter grade are:\n" );
49 printf( "A: %d\n", aCount );
50 printf( "B: %d\n", bCount );
51 printf( "C: %d\n", cCount );
52 printf( "D: %d\n", dCount );
53 printf( "F: %d\n", fCount );
54
55 return 0;
56 }
```



Outline



Program Output

```
Enter the letter grades.  
Enter the EOF character to end input.  
A  
B  
C  
C  
A  
D  
F  
C  
E  
Incorrect letter grade entered. Enter a new grade.  
D  
A  
B  
  
Totals for each letter grade are:  
A: 3  
B: 2  
C: 3  
D: 2  
F: 1
```



Outline



1. Initialize variable

2. Loop

3. Print

```
1  /* Fig. 4.9: fig04_09.c
2     Using the do/while repetition structure */
3  #include <stdio.h>
4
5  int main()
6  {
7     int counter = 1;
8
9     do {
10        printf( "%d ", counter );
11    } while ( ++counter <= 10 );
12
13    return 0;
14 }
```

1 2 3 4 5 6 7 8 9 10

Program Output

La struttura di iterazione **for**

Pascal

```
for variabile := espr2 to espr3 do  
    istruzione
```

C

```
for (espr1; espr2; espr3)  
    istruzione
```



Outline



4.6 Examples Using the for Structure

Program to sum the even numbers from 2 to 100

Program Output

```
1  /* Fig. 4.5: fig04_05.c
2     Summation with for */
3  #include <stdio.h>
4
5  int main()
6  {
7     int sum = 0, number;
8
9     for ( number = 2; number <= 100; number += 2 )
10        sum += number;
11
12    printf( "Sum is %d\n", sum );
13
14    return 0;
15 }
```

Sum is 2550

La prossima volta ...

- Struttura di un programma C
- Funzioni
- I/O base
- Puntatori
- Tipi derivati (vettori, strutture)
- Introduzione all'ambiente Borland C++