

Esercizi di preparazione all'esame

Mauro Brunato

16 maggio 2006

Esercizio 1

Scrivere una funzione `media_armonica(...)` che restituisca la media armonica degli elementi di un vettore `double`. La media armonica \bar{x} di un vettore $(x_i)_{i=0, \dots, n-1}$ è definita come segue:

$$\bar{x} = \frac{n}{\sum_{i=0}^{n-1} \frac{1}{x_i}}$$

Esercizio 2

2.1) Scrivere una funzione che, data una stringa di caratteri, sostituisca ogni lettera 'a' minuscola in essa presente con una lettera 'e' minuscola.

2.2) Scrivere un programma che chieda una parola all'utente, la modifichi chiamando la funzione richiesta in 2.1 e scriva il risultato della modifica sullo schermo.

Esercizio 3

3.1) Come funziona e cosa fa la seguente funzione?

```
int gulp (int x[], int L)
{
    int i;
    for ( i = 1; i < L; i++ )
        x[i] = x[i] + x[i-1];
    return x[L-1];
}
```

3.2) Descriverne il funzionamento per $L = 4$ e $x =$

5	7	9	8
---	---	---	---

.

Esercizio 4

Scrivere un programma che chieda all'utente di inserire un numero arbitrario di interi, li memorizzi in un vettore intercalati con degli zeri e stampi il contenuto finale del vettore, come nel seguente esempio (i dati inseriti dall'utente sono in neretto):

Quanti numeri? 5
Inserire 5 numeri: 7 9 1 4 2
Contenuto del vettore: 7 0 9 0 1 0 4 0 2 0

Esercizio 5

5.1) Elencare i quattro errori presenti nella seguente funzione (considerare errori di sintassi, errori di concetto, istruzioni inutili). Notare che, in particolare, non è definita alcuna variabile. Riscrivere infine la funzione correttamente.

```
char mistero (char s[])
{
    i == 0;
    while ( s[i] > 'i' )
        i = i + 1
    return s(i);
}
```

5.2) Che cosa restituisce la funzione `mistero()`, una volta corretta, se il parametro `s` è la stringa “pippo”? E se vale “pluto”?

Suggerimento — *Il codice ASCII del terminatore di stringa '\0' è 0, inferiore a tutti gli altri caratteri.*

Esercizio 6

6.1) Scrivere una funzione `media(...)` che restituisca la media di un insieme di numeri `double` contenuti in un vettore. Tale funzione ha come parametri il vettore e la sua dimensione.

6.2) Scrivere un programma principale che riempie un vettore con dati inseriti dall'utente e stampa la loro media utilizzando la funzione scritta al punto 6.1.

Esercizio 7

7.1) Spiegare che cosa fa la seguente funzione (dove `strlen(s)` restituisce la lunghezza della stringa `s`):

```
void ignota (char pippo[])
{
    int ciccio, i, j;

    for ( i = 0, j = strlen(pippo)-1; i < j; i++, j-- ) {
        ciccio = pippo[i];
        pippo[i] = pippo[j];
        pippo[j] = ciccio;
    }
}
```

7.2) Spiegare la funzione in seguito alla seguente modifica:

```
void ignota (char pippo[])
{
    int ciccio, i, j;

    for ( i = 0, j = strlen(pippo)-1; i < j; i++, j-- )
        pippo[i] = pippo[j];
}
```

Esercizio 8

8.1) Realizzare un programma che chieda all'utente un numero arbitrario di interi da collocare in un vettore. Si eseguano i controlli su possibili errori (es: dimensione insufficiente del vettore).

8.2) Sviluppare una funzione che abbia come parametri un vettore e la sua dimensione e che sostituisca tutti i “77” con dei “99”.

8.3) Si esegua un test della funzione usando il programma sviluppato nel punto 8.1.

Esercizio 9

Sviluppare una funzione `aumenta_di_tre(...)` che incrementi di 3 il valore di una variabile intera passata come argomento.

Esercizio 10

10.1) Descrivere il funzionamento della seguente funzione:

```
int checosafaccio (char coso[3])
{
    int i, pippo;
    pippo = 0;
    for ( i = 0; i < 3; i++ )
        pippo = pippo*10 + (coso[i] - '0');
    return pippo;
}
```

In particolare, dire qual è il valore restituito dalla funzione applicata alla stringa

'7'	'9'	'2'
-----	-----	-----

10.2) Descrivere il funzionamento della seguente modifica alla funzione:

```
int checosafaccio (char coso[])
{
    int i, pippo;
    pippo = 0;
    for ( i = 0; coso[i] != '\0'; i++ )
        pippo = pippo*10 + (coso[i] - '0');
    return pippo;
}
```

In particolare, qual è il risultato se applichiamo la funzione alla stringa

'6'	'0'	'\0'
-----	-----	------

Esercizio 11

11.1) Scrivere una funzione `raddoppia(...)` che, dato un vettore di tre elementi interi, li raddoppi, come nell'esempio seguente:

7	3	4
---	---	---

 \Rightarrow

14	6	8
----	---	---

La funzione dovrà accettare come parametro il vettore.

11.2) Scrivere la funzione `prodotto(...)` (modificando la precedente) che, dato un vettore di interi e un numero intero moltiplichi ogni elemento del vettore per l'intero. La funzione dovrà accettare come parametri il vettore, la lunghezza del vettore e il numero da moltiplicare.

Esercizio 12

12.1) Utilizzando esclusivamente operazioni aritmetiche su interi e cicli iterativi, sviluppare una funzione `due_alla(...)` che restituisca 2^n , dove n è il valore di una variabile intera passata come argomento.

12.2) Inserire la funzione `due_alla(...)` in un programma funzionante, che chieda all'utente l'inserimento di un numero intero e stampi il risultato dell'elevamento a potenza.

Esercizio 13

13.1) Scrivere la funzione `cerca(v, n, L)` che accetta come parametri un vettore di interi `v`, un intero `n` da cercare nel vettore e la lunghezza `L` del vettore e restituisce `1` se il numero `n` è presente nel vettore `v`, `0` altrimenti.

13.2) Modificare la funzione `cerca(v, n, L)` affinché restituisca l'indice dell'elemento di `v` uguale a `n`, se esiste, altrimenti il numero `-1`.

Esercizio 14

14.1) Cosa stamperà il seguente programma?

```
#include <stdio.h>

void whatisthis (int *a, int *b)
{
    if ( *a < *b )
        *a = *b;
    else
        *b = *a;
}

int main (void)
{
    int x, y;
    x = 7;
    y = 10;
    whatisthis (&x, &y);
    printf ("%d, %d\n", x, y);
    return 0;
}
```

14.2) A cosa serve in generale la funzione `whatisthis()`? Cosa cambierebbe se invece di passare come parametri i puntatori si passassero direttamente le variabili, cioè se la funzione fosse definita come `whatisthis (int a, int b)`, togliendo tutti gli asterischi dal corpo e dall'intestazione della funzione e chiamandola da `main()` con il comando `whatisthis (x, y)`?

Esercizio 15

Definire una funzione

```
int minimo (int v[], int p, int n)
```

che, dato un vettore `v` contenente `n` interi e un indice `p = 0, ..., n - 1`, restituisce l'indice del minimo intero contenuto nel vettore `v` a partire dalla posizione `p` fino alla fine (posizione `n - 1`). Ad esempio, se

$v =$

2	4	6	5	9	10	7
---	---	---	---	---	----	---

,

`minimo (v, 2, 7)` vale `3`. Infatti, `3` è l'indice dell'elemento di valore `5`, ovvero il più piccolo di `v` a partire dalla posizione `2`.

Esercizio 16

16.1) Scrivere la funzione

```
int quanti_minori (int v[], int n, int x)
```

che, dato il vettore di interi v di n elementi e un numero intero x , restituisca il numero di elementi di v *strettamente minori* di x . Per esempio, se

```
v = 

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 5 | 3 | 2 | 7 | 9 | 4 |
|---|---|---|---|---|---|

,
```

allora `quanti_minori (v, 6, 5)` vale 3.

16.2) Utilizzando la funzione `quanti_minori()`, scrivere la funzione

```
void copia_ordinato (int v[], int w[], int n),
```

dove v è un vettore di n interi *distinti*. La funzione dovrà copiare gli elementi di v nel vettore w *riordinandoli* secondo il seguente algoritmo:

- Per ogni elemento $v[i]$ ($i = 0, \dots, n - 1$):
 - Contare il numero k di elementi di v che sono strettamente minori di $v[i]$;
 - Porre l'elemento $v[i]$ alla posizione k -esima di w .

Il vettore v non dovrà essere modificato e la funzione non dovrà restituire alcun valore; l'eventuale contenuto iniziale di w non dovrà influire in alcun modo sul funzionamento.

Esercizio 17

17.1) Descrivere il funzionamento della seguente funzione:

```
int wasistdas (double v[], double w[], int n)
{
    int i, j;
    j = 0;
    for ( i = 0; i < n; i++ )
        if ( v[i] > 2.0 ) {
            w[j] = v[i];
            j++;
        }
    return j;
}
```

17.2) Come si comporta la funzione `wasistdas()` passandole il vettore

```
v = 

|     |     |     |     |
|-----|-----|-----|-----|
| 5.0 | 3.2 | 2.2 | 1.4 |
|-----|-----|-----|-----|


```

di lunghezza 4?

Esercizio 18

Cosa stamperà il seguente programma?

```
#include <stdio.h>

int inutile (int a, int *b, int *c)
{
    *b = *c;  b = &a;
    *c = *b;  a = *c;
    return a;
}

int main (void)
{
    int a = 1, b = 2, c = 3;
    int d = inutile (a, &b, &c);
    printf ("%d %d %d %d\n", a, b, c, d);
    return 0;
}
```

Esercizio 19

Scrivere la funzione

```
void raddoppia (char a[])
```

che raddoppia il contenuto della stringa a ponendo il contenuto della stringa in coda alla stessa, come dal seguente esempio.

Se

```
a = [ 'P' | 'i' | 'p' | 'p' | 'o' | '\0' |   |   |   |   |   |   ]
```

dopo la chiamata `raddoppia(a)` la stringa a contiene la seguente stringa:

```
a = [ 'P' | 'i' | 'p' | 'p' | 'o' | 'P' | 'i' | 'p' | 'p' | 'o' | '\0' |   ]
```

Esercizio 20

Scrivere la funzione

```
int palindromo (char[])
```

che restituisce 1 se la stringa passata come parametro è palindroma, 0 altrimenti. Ricordiamo che una stringa si dice *palindroma* se non cambia invertendo l'ordine delle lettere (ad esempio, "radar"). In altri termini, il primo e l'ultimo carattere sono uguali, così pure il secondo e il penultimo e così via.

Esercizio 21

Descrivere il funzionamento della seguente funzione:

```
int fuffi (int v[], int L)
{
    int qui, quo, qua;
    qui = 0;
    for ( quo = 0; quo < L; quo++ ) {
        for ( qua = 0; qua < v[quo]; qua++ )
            printf ("X");
        printf ("\n");
        qui = qui + v[quo];
    }
    return qui;
}
```

In particolare, descrivere il comportamento della funzione (output e valore restituito) sul vettore

$v =$

3	5	4	2	6
---	---	---	---	---

di lunghezza $L = 5$.

Esercizio 22

Sia v definito nel seguente modo:

```
char *v = "Zaphod Beeblebrox";
```

Due tra le espressioni che seguono sono errate, le altre sono equivalenti a due a due. Individuare le espressioni errate (spiegando perché vanno ritenute tali) e accoppiare le altre in base alla loro equivalenza.

$*v$	$\&(v+1)$	$\&(\&(v[3])-1)$	v
$v[2]$	$\&(v[0])-v$	$\&(v[0])$	$\&(v-1)$
$v+2$	$v[0]$	0	$\&(v[3])-1$

Esercizio 23

Scrivere la funzione

```
void rovescia (char *dest, char *src)
```

che copia la stringa src in $dest$ rovesciandola. Se v e w sono due vettori di caratteri e v contiene la stringa "pippo", dopo la chiamata

```
rovescia (w, v)
```

la stringa contenuta in v deve restare invariata, mentre w deve contenere la stringa "oppip". Verificare che i terminatori siano trattati correttamente.

Per gestire il colore, il computer ne considera la scomposizione nei tre colori, detti *primari*, a cui i coni della retina umana sono sensibili: il rosso, il verde e il blu. Questa scomposizione è detta *RGB*, dalle iniziali dei nomi inglesi dei tre colori primari. Ogni colore può essere rappresentato dunque da un vettore di tre numeri reali compresi tra 0 e 1, dove 0 rappresenta l'assenza della componente e 1 l'intensità massima sostenibile dal dispositivo di visualizzazione. Per esempio, il vettore

0.8	0.8	0.2
-----	-----	-----

rappresenta un arancione chiaro (componenti rossa e verde quasi al massimo, un po' di blu). L'intensità luminosa di un colore è una media dell'intensità delle sue tre componenti RGB, pesata in particolare sul verde, a cui l'occhio umano è maggiormente sensibile.

Esercizio 24

Supponendo che il verde influisca sulla luminosità il doppio delle altre due componenti, scrivere la funzione

```
float intensita (float colore[3])
```

che restituisce l'intensità luminosa del colore le cui componenti RGB sono memorizzate nel vettore reale `colore`. Se ad esempio il vettore `colore` contiene le componenti scritte sopra,

$$\text{intensita}(\text{colore}) = \frac{0.8 + \overbrace{0.8 + 0.8}^{\text{componente verde}} + 0.2}{4} = 0.65.$$

Esercizio 25

Scrivere la funzione

```
void combina (double A[10][10], double alfa, int n)
```

dove A è una matrice reale quadrata di dimensione n (al più 10) e alfa è un numero reale. La funzione deve sommare a tutte le righe di A , tranne la prima, la prima riga moltiplicata per il coefficiente alfa . Se, ad esempio,

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 8 & 12 \end{pmatrix},$$

dopo la chiamata "`combina (A, -2, 3);`" la matrice A deve valere

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \\ 2 & 4 & 6 \end{pmatrix}.$$

Il problema dell'anno 2000, ormai noto a tutti, nasce dal fatto che molti programmi scritti prima del 2000 memorizzavano le sole due cifre meno significative dell'anno, dando per scontato che le cifre secolari fossero '19', rendendo così impossibile rappresentare l'anno 2000 e i successivi. Un metodo per correggere l'errore continuando a usare numeri di due cifre per rappresentare l'anno è il seguente *metodo a soglia*: se il numero composto dalle due cifre è maggiore di 60, allora esso rappresenta una data del XX secolo, se vale 0 rappresenta il 2000, se è minore o uguale a 60 rappresenta un anno del XXI secolo. Ad esempio, 89 rappresenta l'anno 1989, mentre 36 rappresenta l'anno 2036. In questo modo si possono rappresentare gli anni dal 1961 al 2060.

Esercizio 26

Scrivere una funzione

```
int confronta2000 (int a, int b)
```

dove a e b sono due numeri compresi fra 0 e 99 che restituisce 1 se a rappresenta un anno maggiore di quello rappresentato da b secondo il metodo a soglia, 0 se sono uguali, -1 se l'anno rappresentato da a è minore. Ad esempio, `confronta2000(80,41)` vale -1 , mentre `confronta2000(80,67)` vale 1.

Esercizio 27

Elencare i due errori sintattici del seguente programma:

```
#include <stdio.h>

int main (void)
{
    int i = 0;
    while i < 10
        printf ("%d\n", ++i)
    return 0;
}
```

Quale sequenza di numeri stamperà il programma, una volta corretta la sintassi?

Esercizio 28

Che cosa stamperà il seguente programma e perché?

```
#include <stdio.h>

int main ()
{
    char *v = "Non sono felice.";
    int i;
    printf ("%s\n", v);
    for ( i = 0; v[i] != ' '; i++ )
        printf ("e %d...\n", i);
    printf ("Ora%s\n", v+i);
    return 0;
}
```

Esercizio 29

Ricordando che l'errore di troncamento di una serie convergente a termini di segno alterno è maggiorato dal valore assoluto del primo termine non considerato (teorema di Leibniz), scrivere un programma che calcoli il valore della serie

$$\sum_{n=0}^{+\infty} \frac{(-1)^n}{n^2 + 1}$$

con un errore massimo di 10^{-5} .

Suggerimento — ottenere n^2 moltiplicando per se stessa la variabile intera n ; per evitare che la frazione venga considerata intera, imporre che le costanti siano reali scrivendo `1.0` al posto di `1`; per calcolare il segno del termine non è necessario elevare a potenza, basta una semplice condizione sulla parità di n .

Secondo il calendario giuliano, in vigore fino al 1581, un anno era bisestile se e solo se divisibile per quattro. A partire dalla riforma del calendario operata da papa Gregorio XIII nel 1582, gli anni bisestili sono tutti e soltanto quelli divisibili per quattro, tranne gli anni secolari a meno che non siano divisibili per quattrocento; per esempio, il 2000 è stato bisestile, mentre il 1900 non lo è stato. Il 1500, appartenente al vecchio calendario giuliano, era bisestile, anche se non lo sarebbe stato secondo le nuove regole.

Esercizio 30

30.1) Scrivere la funzione

```
int bisestile (int anno)
```

che restituisce 1 se l'anno passato a parametro è bisestile, 0 altrimenti.

30.2) Scrivere un programma che chieda un anno e, utilizzando la funzione appena scritta, scriva se l'anno è bisestile o no.

Suggerimento — *Per calcolare la divisibilità utilizzare l'operatore di resto.*

Esercizio 31

Scrivere un programma che definisce una matrice reale A di dimensione 4×4 , la inizializza come matrice identità

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

e ne stampa il contenuto ordinato per righe e colonne, come nel seguente esempio:

```
1.0000 0.0000 0.0000 0.0000
0.0000 1.0000 0.0000 0.0000
0.0000 0.0000 1.0000 0.0000
0.0000 0.0000 0.0000 1.0000
```

Suggerimento — *Riempire la matrice di zeri con due cicli annidati, poi modificare gli elementi della diagonale con un singolo ciclo. Nella stampa, ricordarsi di aggiungere un accapo al termine di ogni riga. Usare il descrittore "%6.4f " per stampare un numero reale secondo il formato richiesto (notare lo spazio).*

Esercizio 32

Cosa stamperà il seguente programma? E perché?

```
#include <stdio.h>

int main ()
{
    int a = 1, b = 17;
    char c = 'a';
    float d = 154e-2;

    c += b % ( (d<a)?5:6 );
    if ( c > 'd' )
        printf ("Che programma stupido!\n");
    else
        printf ("Che programma inutile!\n");
}
```

Suggerimento — *I caratteri sono memorizzati e trattati nelle espressioni secondo il loro codice ASCII. I codici ASCII delle lettere minuscole sono adiacenti e seguono il normale ordine alfabetico inglese. Non è necessario conoscere il codice ASCII per risolvere l'esercizio e basta ricordare l'alfabeto italiano.*

Esercizio 33

Scrivere un programma che per ogni intero i da 2 a 10 stampi il prodotto dei numeri pari da 2 a i .

Suggerimento — *Il programma deve stampare i seguenti numeri: 2, 2, 8, 8, 48, 48, 384, 384, 3840.*

Esercizio 34

Scrivere un programma che per ogni numero intero i da 1 a 10 stampi tutti i numeri da 1 a i per cui i è divisibile.

Suggerimento — *Il programma deve stampare le seguenti linee:*

```
1
1 2
1 3
1 2 4
1 5
1 2 3 6
1 7
1 2 4 8
1 3 9
1 2 5 10
```

Esercizio 35

Che cosa scriverà il seguente programma?

```
int f (int b)
{
    int c;
    c = b;
    return c + b;
}
int main (void)
{
    int a = 1;
    while ( a < 4 )
        printf ("%d\n", f(a++));
    return 0;
}
```

Esercizio 36

36.1) Scrivere una funzione che accetti due puntatori a interi p e q , confronti i valori contenuti agli indirizzi cui p e q puntano e sostituisca il valore più grande col più piccolo.

36.2) Scrivere un programma principale che utilizza questa funzione.

Esercizio 37

Che cosa scrive la seguente istruzione per diversi valori di n ?

```
printf ("Oggi sono %s.\n", "infelice" + 2 * (n%2));
```

Suggerimento — *Provare per $n=0$ e $n=1$, poi ricavare una regola generale.*

Esercizio 38

38.1) Siano date le tre variabili intere $i=2$, $a=3$ e $b=4$; Dire quanto vale la seguente espressione, e i valori delle tre variabili dopo la sua valutazione:

```
a++ == b && ++i == 3
```

38.2) Dire quanto valgono le seguenti espressioni, e i valori delle tre variabili dopo ciascuna di esse:

```
a++ == b || ++i == 3
++a == b || ++i == 3
```

Considerare, prima di ciascuna espressione, i valori *iniziali* delle tre variabili, come se ciascuna formula fosse eseguita indipendentemente dalle altre.

Suggerimento — *Ricordare le regole relative alla “valutazione pigra”!!*

Esercizio 39

Utilizzando gli operatori logici sui bit, scrivere la funzione

```
int primouno (int x)
```

che restituisce l'indice del primo bit di x uguale a 1 a partire dal meno significativo, dove il bit meno significativo ha indice 0. Ad esempio, `primouno(16)` vale 4, perché $16_{10} = 10000_2$, e il primo 1 occupa la posizione numero 4 a partire dalla cifra meno significativa. Si assuma che x sia positivo.

Suggerimento — *Contare il numero di scorrimenti a destra da applicare al numero prima di ottenere un numero dispari.*

Un vettore $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ si dice *supercrescente* se ogni suo elemento è maggiore della somma di quelli che lo precedono:

$$\forall i = 1, \dots, n \quad x_i > \sum_{j=1}^{i-1} x_j.$$

Considerato nullo il valore della sommatoria vuota, è chiaro che il primo termine dev'essere positivo, e così tutti gli altri per induzione. È altrettanto chiaro che un vettore supercrescente è ordinato.

Esercizio 40

40.1) Scrivere una funzione

```
int supercrescente (double v[], int n)
```

che accetti come parametro un vettore v di numeri `double` e un intero n rappresentante il numero di elementi del vettore. La funzione restituisce vero se e solo se il vettore v di n elementi è supercrescente.

40.2) Utilizzando ove opportuno una chiamata alla funzione definita al punto 40.1, scrivere una funzione `main` che chiede la lunghezza di un vettore di `double`, lo alloca dinamicamente, lo riempie di valori letti al terminale, dice se è supercrescente oppure no, infine libera la memoria.

Esercizio 41

41.1) Supponiamo per semplicità che due parole realizzino una *rima* se e solo se hanno almeno gli ultimi tre caratteri uguali. Scrivere una funzione che accetta a parametro due stringhe e che restituisce vero se e solo se le parole contenute nelle due stringhe rimano.

41.2) Scrivere un programma principale che utilizza la funzione scritta al punto 41.1 per dire all'utente se due parole da lui inserite rimano o no.

Esercizio 42

Scrivere la funzione

```
void enumera (int v[], int n, int c[], int max)
```

che riceve in ingresso un vettore v , contenente n elementi di valore compreso tra 0 e max , e il vettore non inizializzato c , di $\text{max}+1$ elementi. La funzione dovrà registrare in c il numero di occorrenze di ogni numero tra 0 e max nel vettore v . In altri termini, per ogni i tra 0 e max , $c[i]$ deve contenere il numero di volte che il valore i compare nel vettore v .

Esistono vari modi di generalizzare la successione di Fibonacci. Uno di questi è il seguente.

La *successione di Fibonacci di ordine n* è la successione a valori naturali i cui primi n termini sono uguali a 1, e ogni termine successivo è uguale alla somma degli n immediatamente precedenti.

La successione classica di Fibonacci si ottiene ponendo $n = 2$.

Esercizio 43

43.1) Scrivere la funzione

```
void fibonacci (int n, int c, int *v)
```

che riempie il vettore v con i primi c termini della successione di Fibonacci di ordine n . Si supponga che il vettore abbia lo spazio necessario.

43.2) Scrivere un programma che chiede all'utente l'ordine della successione, il numero di termini da stampare e, dopo aver invocato la funzione di cui al punto precedente, stampi il vettore risultante. Il vettore deve essere allocato dal programma principale.

Esercizio 44

Che cosa scriverà a video il seguente programma?

```
#include <stdio.h>

int modifica (int s, int *p)
{
    *p += p[s];
    return *p;
}

int main (void)
{
    int p[] = {1, 5, 4, 3, 5};
    int i;

    for ( i = 3; i >= 0; i-- )
        printf ("%d %d\n", i, modifica(i,p));
    return 0;
}
```

Esercizio 45

45.1) Sia a una variabile intera. Quale valore bisogna assegnare ad a affinché la seguente espressione

```
((8 >> a > a) ? --a : a--) ? "falso" : "vero"
```

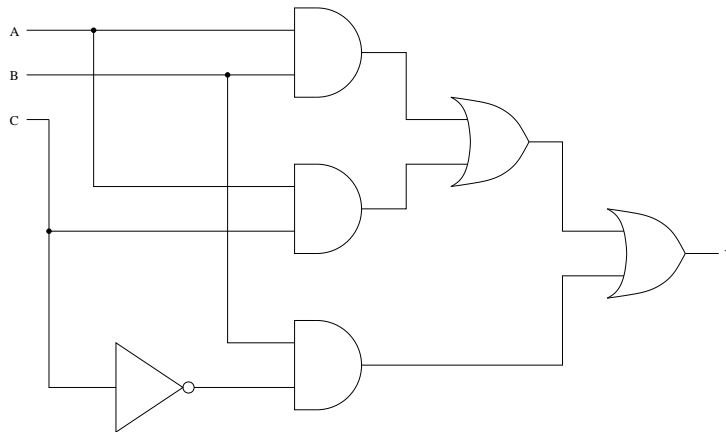
valga "vero"?

45.2) Dimostrare che per ogni altro valore di a l'espressione vale "falso".

Suggerimento — (per il primo punto) provare con valori piccoli di a , oppure ragionare prima sul secondo punto.

Esercizio 46

46.1) Dato il circuito logico di figura, composto da porte “and”, “or” e “not”:



indicare l'espressione logica corrispondente che lega la variabile Y alle variabili logiche A , B e C .

46.2) Calcolare la tabella di verità dell'espressione così ricavata.

Esercizio 47

47.1) Scrivere la tabella di verità dell'espressione logica

$$Y = (A \wedge B) \vee (A \wedge \bar{C}) \vee (\bar{B} \wedge C)$$

dove A , B e C , sono variabili logiche (aventi valore vero o falso).

Ricordiamo che, nella notazione logica standard, “ \wedge ” è l'operatore di congiunzione (“e”), equivalente a “&&” in C, “ \vee ” è l'operatore di disgiunzione (“o”), equivalente a “| |” in C, mentre la barra rappresenta la negazione logica (“not”), corrispondente a “!” in C.

47.2) Disegnare il circuito a porte logiche corrispondente all'espressione logica.