

COMPILARE PASCAL

Settembre 2002

Enrico Centenaro enrico@matematiche.org

E' possibile copiare e modificare liberamente questo documento secondo le indicazioni della licenza GPL (vedi www.gnu.org).

Indice

1	Iniziamo	3
2	A me interessa il Pascal	3
2.1	Il Turbo Pascal	4
2.2	Free Pascal	4
2.3	Il Pascal di Delphi	4
2.4	E gli altri	5
3	Il Turbo Pascal 5.5	5
3.1	Installazione	5
3.2	Istruzioni di utilizzo	5
3.2.1	L'Ambiente di sviluppo programmi	6
3.3	Realizzazione di un programma	7
3.3.1	Apertura della finestra di editor	7
3.3.2	Scrittura del programma	7
3.3.3	Compilazione del programma	7
3.3.4	Esecuzione del programma	7
3.3.5	Creazione di una versione eseguibile del programma	7
3.4	Individuazione degli errori	8
3.4.1	La pila delle attivazioni	8
3.4.2	Esecuzione del programma istruzione per istruzione	8
3.4.3	Ispezione delle variabili	9
3.4.4	Punti di arresto	10
3.5	Uso di file di dati esterni	10
3.6	I comandi dei menù	12
3.6.1	Menù File	12
3.6.2	Menù Edit	12
3.6.3	Menù Search	13
3.6.4	Menù Run	14
3.6.5	Menù Debug	14
3.6.6	Menù window	14
3.7	Librerie di sottoprogrammi (unit)	15
3.7.1	Uso delle unit standard	15
3.7.2	Costruire una unit	17
3.7.3	Compilazione di una unit	20
3.7.4	Funzionalita' Della Unit	20
3.7.5	Uso della unit	20
4	Free Pascal	20
4.1	Installazione	21
4.2	Uso del compilatore	21
4.3	Opzioni del compilatore	22

COMPILARE PASCAL *

Enrico Centenaro

2002

Sommario

Il Pascal è un esempio di linguaggio di programmazione ad alto livello. Ci sono molti altri linguaggi di programmazione e ognuno ha i suoi vantaggi e svantaggi.

L'animo di questo scritto è quello fare una panoramica sui principali compilatori Pascal . Dato che questa dispensa è rivolta a studenti 'alle prime armi', non c'è la pretesa di trattare 'tutto', ma fornire delle indicazioni che possano essere l'inizio di un percorso.

Ci soffermeremo su due compilatori che ritengo particolarmente interessanti dal punto di vista didattico: il Turbo Pascal e il più recente Free Pascal, sono molto simili fra loro, ma si fondano su filosofie diverse.

1 Iniziamo

I compilatori sono fondamentali per la moderna informatica. Essi agiscono come traduttori, trasformando i linguaggi di programmazione orientati all'uomo (human oriented) nei linguaggi di programmazione orientati alla macchina (machine oriented).

Volendo essere più precisi, il compilatore è un applicativo in grado tradurre un testo, scritto in un linguaggio, in codice macchina. Tutti gli applicativi, a prescindere dal linguaggio con cui sono scritti, per essere eseguiti su di un elaboratore devono essere convertiti in codice macchina o, più semplicemente, in Assembly. Lo scopo di un compilatore è quindi quello di tradurre delle sigle in numeri. Esistono, infatti, compilatori C, compilatori Pascal, compilatori Fortran, ecc... Un linguaggio è sempre legato ad un compilatore e un compilatore è legato ad una macchina. Questo significa che per poter sviluppare in Pascal , non basta dotarsi di un compilatore Pascal qualsiasi ma, bensì, di un compilatore Pascal per la macchina - e sistema operativo - su cui dovrà essere eseguito il nostro applicativo.

2 A me interessa il Pascal

Il Pascal è una creazione di Wirth. Il linguaggio Pascal fu in origine progettato da Niklaus Wirth, un professore svizzero, alla fine degli anni '70.

Quando il Pascal fu progettato, esistevano già molti altri linguaggi di programmazione, ma pochi largamente usati: FORTRAN, C, Assembly, COBOL, ADA, ALGOL. L'idea chiave del nuovo linguaggio di programmazione fu l'ordine, raggiunto tramite un forte concetto di tipizzazione dei dati, il controllo delle dichiarazioni e la programmazione strutturata. Il linguaggio fu anche sviluppato come strumento di insegnamento nelle scuole.

*Questo documento è stato scritto utilizzando Latex www.latex-project.org (vedi [4] e [5])

Una cosa è l'ideazione e un'altra è l'implementazione. Wirth ha inventato il linguaggio, ma altri ne hanno scritto i compilatori. Riportiamo una panoramica di quelli più noti.

2.1 Il Turbo Pascal

Il famoso compilatore Pascal di Borland, chiamato Turbo Pascal, fu introdotto nel 1985. Il compilatore Turbo Pascal fu un best-seller dell'epoca e rese il linguaggio popolare particolarmente nella piattaforma PC, grazie anche all'unione di semplicità e potenza.

Il Turbo Pascal introdusse un ambiente di sviluppo integrato (IDE) in cui era possibile editare il codice (in un editor WordStar¹ compatibile), compilare, vedere gli errori e saltare alle linee contenenti gli stessi. Oggi questo può sembrare banale e scontato, ma in precedenza si doveva uscire dall'editor tornando al DOS, compilare, annotarsi le righe di errore e aprire di nuovo l'editor puntando alle righe incriminate.

Inoltre Borland vendeva Turbo Pascal per 49 dollari, dove Microsoft vendeva il suo compilatore Pascal per alcune centinaia di dollari. I molti anni del successo del Turbo Pascal contribuirono alle decisioni di Microsoft di cancellare la linea dei suoi compilatori Pascal.

2.2 Free Pascal

Il compilatore freepascal è un ottimo compilatore freeware multiplatforma, infatti il FP è disponibile per diversi sistemi operativi tra cui Unix/Linux e Dos/Windows.

Il progetto Free Pascal contiene un IDE (Interface Environment Editor) che si chiama FPIDE molto simile a quello del Turbo Pascal, ma vi sono degli altri prodotti altrettanto gratuiti che forniscono una interfaccia grafica al compilatore FPC.

Troviamo FPE che nell'ambiente Windows è molto usata e gli stessi sviluppatori di FP la consigliano.

Dev-pascal è un altro IDE gratuito disponibile per FP, supporta tutto quello che un IDE dovrebbe supportare, debugging compreso.

Lazarus è un progetto che ha lo scopo di realizzare un clone del Delphi basato però su un compilatore gratuito. Questo progetto include una interfaccia grafica, una sua libreria chiamata LCL che si interfaccia con le librerie GTK+ sia in Linux che Windows.

MOS (Master Operating System) una GUI (Graphic User Interface) simile al Visual Pascal per realizzare programmi 'visuali' che supporta Borland Pascal 7 e Free Pascal; viene fornita con uno strumento di sviluppo visuale per l'editing dei programmi. E' disponibile per piattaforme DOS e Windows.

2.3 Il Pascal di Delphi

Dopo 9 versioni dei compilatori Turbo e Borland Pascal, che gradualmente estesero il linguaggio, Borland rilasciò nel 1995 Delphi, rendendo così il Pascal un linguaggio di programmazione visuale.

Delphi estende il linguaggio Pascal in diversi punti, incluso diverse estensioni object-oriented, che differiscono da altre versioni dell'Object Pascal, incluso quelle del compilatore Borland Pascal with Objects.

¹Celeberrimo word processor, c'era quando Bill Gates scriveva i suoi primi programmi, successivamente ha fatto i soldi...

2.4 E gli altri

- Irie Pascal disponibile per diverse piattaforme
- TMT Pascal Lite (windows)
- DPas (windows)
- Bloodshed DevPascal (windows) (non è solo un compilatore ma un vero e proprio ambiente di sviluppo)
- GNU Pascal (linux/unix)

3 Il Turbo Pascal 5.5

3.1 Installazione

Il compilatore Pascal viene distribuito gratuitamente e lo si può scaricare dal sito <http://community.borland.com> (vedere alla voce 'Museum').

Suggerimenti sulla Procedura di Installazione: Dal momento che la procedura di installazione del Turbo Pascal è abbastanza obsoleta, è opportuno in fase di installazione seguire i seguenti suggerimenti:

- Decomprimere il contenuto del file TurboPascal55.zip in una cartella del disco rigido (Es: C:\appoggio\TP); verranno create due sottocartelle separate chiamate disk1 e disk2.
- Copiare il contenuto della cartella disk2 all'interno di disk1, in modo da avere tutti i file necessari per l'installazione all'interno di un'unica cartella.
- Eseguire la procedura di installazione install.exe una prima volta; verrà restituito un elenco di messaggi di errore; chiedere di ignorare gli errori e procedere fino al termine dell'installazione.
- Eseguire una seconda volta la procedura di installazione install.exe; questa volta non dovrebbero essere segnalati ulteriori errori e l'installazione dovrebbe andare a buon fine.

Aggiornamento del Sistema Operativo: dopo aver eseguito l'installazione, per fare in modo che il compilatore (TPC.exe) sia eseguibile da qualsiasi cartella, è necessario aggiornare la variabile d'ambiente PATH, aggiungendo tra i percorsi specificati il percorso corrispondente alla cartella in cui è installato il Turbo Pascal (normalmente C:

TP); chi conosce la procedura necessaria può aggiornare la variabile PATH una volta per tutte (in ambiente Windows'95-'98, questo richiede di modificare il file C:

AUTOEXEC.BAT). Chi non conosce la procedura necessaria, può modificare il valore della variabile all'inizio di ogni sessione di lavoro con il compilatore; per farlo, dopo aver lanciato il Prompt di MS-DOS, basta digitare, come primo comando:

```
SET PATH=%PATH%;C:\TP; <invio>
```

3.2 Istruzioni di utilizzo

Dall'ambiente Dos del computer che avete in dotazione digitate i seguenti comandi:

```
C:\> cd \TP (per spostarsi nel direttorio del TurboPascal)
```

```
C:\TP> turbo (per eseguire il TurboPascal)
```

3.2.1 L'Ambiente di sviluppo programmi

La schermata che compare è la seguente:

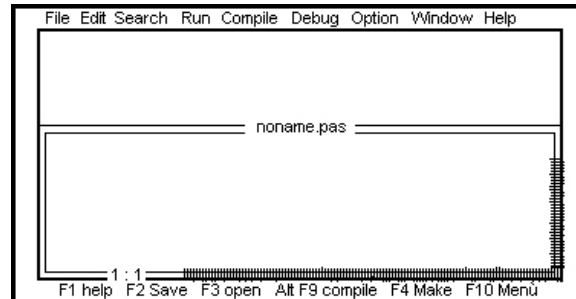


Figura 1: Una tipica schermata del Turbo Pascal

La linea al fondo dello schermo indica alcuni comandi principali eseguibili con i tasti **F** (**F1...F10**). Tra questi il comando **F1** per richiedere aiuto (help), cioè spiegazioni sulle operazioni eseguibili nel contesto in cui si lavora ed il comando **F10** per spostare il cursore sulla barra dei menù (linea superiore dello schermo). Tramite i menù è possibile selezionare i comandi posizionandosi con le frecce sul comando desiderato e premendo quindi il tasto invio. Ad un comando può inoltre essere associato un tasto **F** oppure la combinazione **<alt>-TASTO F**, oppure una sequenza costituita da **<alt>-tasto** seguito da una lettera.

Se si vuole tornare definitivamente al DOS basta eseguire il comando **exit** del menù file (**<alt>-f** e poi **x**). Se invece si vuole abbandonare temporaneamente l'ambiente TurboPascal si usa il comando **dos shell** del-menù file. Per rientrare in TurboPascal si usa il comando DOS **exit**.

Lo schermo iniziale è diviso in due parti, corrispondenti a due finestre denominate: **<output>** e **<noname00.pas>**. La finestra di output è quella in cui vengono mostrati i risultati dell'esecuzione dei programmi. La finestra **noname00.pas** è quella in cui viene scritto il testo del programma, e si chiama finestra di edit. E' possibile ingrandire e spostare queste finestre, se ne possono inserire di nuove e chiudere quelle già aperte. Per aprire finestre vedi: menù file comando open o new, per spostarle, ingrandirle, chiuderle: menù **window** comando **zoom**, **size move**, **close**. Tutte le finestre che l'utente apre vengono messe una sotto l'altra, e resta visibile di esse solo il nome (**noname00.pas** o **output**) ed il numero di controllo. Il numero di controllo è quel numero che sta a destra del nome della finestra (ad es. **noname00.pas** ha il numero di controllo 1 mentre **output** ha il numero di controllo 2). Solo l'ultima finestra aperta è completamente visibile e tutti i comandi o le funzionalità che si attivano si riferiscono ad essa. Si dice che tale finestra è attiva. Per rendere attiva una finestra si usa **<alt>-n**, con n numero di controllo della finestra che si vuole attivare.

3.3 Realizzazione di un programma

Mostriamo le operazioni principali da effettuare per redigere, compilare, eseguire e memorizzare su disco un programma Pascal.

3.3.1 Apertura della finestra di editor

Si esegue il comando open del menù **File (F3)**, si inserisce nella casella name il nome di un file. In questo modo si crea un file con il nome specificato in cui verrà

memorizzato il testo del programma (vedi comando **save** più avanti). Se il file già esiste in memoria di massa e deve essere modificato, occorre specificarne il nome, per intero se esso si trova in un direttorio diverso da quello corrente. Viene quindi aperta una finestra col nome specificato, in cui compare il testo del programma contenuto nel file.

3.3.2 Scrittura del programma

Per i comandi dell'editor si rimanda al menù **edit** per le operazioni di inserimento e cancellazione del testo ed al menù **search** per la ricerca e sostituzione di parti del testo. Si ricorda che le frecce consentono di posizionare il cursore nel punto sul quale si intende lavorare. Quando si ritiene completato il testo del programma questo viene salvato nel file associato alla finestra tramite il comando **save** del menù **File (F2)**. Se si vuole salvare su un nuovo file il comando è **save as**, che ne richiede il nome; in questo caso il file di partenza rimane inalterato.

3.3.3 Compilazione del programma

Tramite il comando **compile** del menù **compile (<alt>-c e poi c)**. Se ci sono errori in fase di compilazione il cursore si posiziona automaticamente sulla parte di programma in cui è stato commesso l'errore, facilitando così la correzione. Una volta corretto l'errore si deve compilare di nuovo il programma fino a che il compilatore non risponde **compile successful**. A questo punto conviene salvare nuovamente il programma su file.

3.3.4 Esecuzione del programma

Il comando **Run** del menù **Run (<alt>-r seguito da r)** compila il programma e, se non vengono individuati errori, lo esegue. L'input e l'output dei dati avvengono nella finestra di output. Per vedere il risultato dell'esecuzione del programma occorre quindi rendere tale finestra attiva. Per interrompere il programma (ad esempio nei casi di non terminazione) si usano i tasti **<ctrl>-pausa** che devono essere premuti per due volte.

3.3.5 Creazione di una versione eseguibile del programma

Se il programma viene considerato corretto si può costruire una versione eseguibile del programma una versione cioè, che si possa utilizzare senza entrare in TurboPascal mandandola in esecuzione direttamente dal DOS. Menù **compile (<alt-c>)** si digita **d**, la finestra si chiude automaticamente, ma **destination** si è spostato su **disk**. Ciò significa che la prossima volta che si compilerà il programma il compilatore memorizzerà sul disco rigido la versione eseguibile del programma (cioè in linguaggio macchina che è direttamente comprensibile dal calcolatore). A questo punto basta aprire il menù **compile** e premere **c** perchè il compilatore produca la versione eseguibile del programma. Si può verificare che nel direttorio corrente è stato incluso un file col nome con cui si è caratterizzato all'inizio il programma, ed estensione **exe**.

3.4 Individuazione degli errori

Uno degli aspetti più importanti dell'ambiente di programmazione riguarda gli strumenti per l'individuazione degli errori logici contenuti nei programmi. A questo proposito risulta indispensabile utilizzare gli strumenti offerti dall'ambiente di programmazione.

3.4.1 La pila delle attivazioni

La pila delle attivazioni (chiamate) di procedura e funzione può essere molto utile per esaminare l'esecuzione del programma. Per vedere la pila delle chiamate di procedura e funzione si esegue il comando **call-stack** nel menù **Window**. Esso apre una finestra in cui è mostrata la pila delle attivazioni durante l'esecuzione del programma. Si consideri il seguente programma che contiene una procedura ricorsiva per il calcolo del fattoriale.

```

Program esempiostack;
var fatt,n:integer;
  procedure fattoriale(var totale:integer;numero:integer);
  begin
    if numero>1
    then begin
      totale:=numero*totale;
      fattoriale(totale,numero-1)
    end
  end;
begin
  fatt:=1;
  writeln('numero = ');
  read(n);
  fattoriale(fatt,n);
  writeln('Il fattoriale e'' ',fatt)
end.

```

Supponendo che al programma venga fornito il valore 4, si ottiene:

```

FATTORIALE(24,1)

FATTORIALE(24,2)

FATTORIALE(24,3)

FATTORIALE(24,4)

ESEMPIOSTACK

```

tutte le procedure hanno il totale=24 perché il parametro è passato per variabile e tutte le chiamate fanno riferimento alla stessa locazione di memoria.

3.4.2 Esecuzione del programma istruzione per istruzione

Per ottenere una esecuzione istruzione per istruzione del programma occorre eseguire il comando **Trace into** del menù **Run**. Il programma viene compilato e viene posta la barra di esecuzione sull'istruzione **begin**. Ogni volta che si preme **F7** viene eseguita l'istruzione successiva.

Consideriamo il programma seguente che chiede all'utente di inserire una matrice quindi calcola la norma della matrice, cioè la somma dei quadrati dei singoli elementi e poi stampa la matrice e la norma.

```

program sommaN;
var i,m,n: integer;
begin
  somma:=0;

```

```

    readln(n);
    for i:=1 to n do
        begin
            writeln('prossimo dato =');
            readln(m);
            somma :=somma + m
        end;
    writeln(somma)
end.

```

Premendo più volte **F7** si arriva all'istruzione sottolineata. A questo punto vengono chiesti dati di input e, per proseguire, occorre inserirli. Analogamente si può verificare l'effetto delle istruzioni di output.

Il comando **Step over** del menù **Run (F8)**, consente di eseguire in un unico passo l'istruzione corrente, durante l'esecuzione istruzione per istruzione. Con **Step over** quindi, si esegue la chiamata ad una procedura, o funzione, senza cioè seguirne il flusso completo. Ad es.

```

program esempio;
var fatt,numero,i : integer;
    function fattoriale (x : integer) : integer;
        var somma : integer;
        begin
            somma:=1;
            for i:=x downto 1
            do somma:=somma*i;
            fattoriale:=somma;
        end;
begin
    writeln('numero = ');
    readln(numero);
    fatt:=fattoriale(numero);
    writeln('il fattoriale di ',numero,' è: ',fatt);
end.

```

Eseguendo il programma in modo **STEP OVER**, arrivato all'istruzione sottolineata il sistema restituisce il valore del fattoriale senza eseguire la funzione istruzione per istruzione come invece avrebbe fatto con **TRACE INTO**.

3.4.3 Ispezione delle variabili

Durante l'esecuzione istruzione per istruzione è indispensabile verificare il valore delle variabili del programma. A questo scopo occorre specificare le variabili di cui si vuole controllare il valore. Si apre il menù **Debug (ALT<d>)** e si esegue il comando **watches**. Così facendo si attiva un sottomenù nel quale si deve scegliere **add-watches**; **add-watches** chiede le **watch-expression**, cioè il nome delle variabili di cui si vuole controllare il flusso (si può attivare direttamente l'opzione **add-watches** premendo **CRTL-F7**); facendo riferimento al programma precedente può essere utile controllare il valore che assumeranno istruzione dopo istruzione le variabili somma ed n, queste sono le **watch-expression** che vanno inserite. Per vedere il valore di tali variabili occorre attivare la finestra watch, tramite il menù **windows (<alt>-w>)** e si esegue il comando **watch**. In tale finestra compaiono i valori che assumono le variabili specificate in precedenza. Se il programma ancora non è stato mandato in esecuzione nella finestra **Watch** accanto al nome della generica variabile ci sarà: **unknown variable**. Non appena si manda in esecuzione il

programma le variabili assumono i valori determinati dall'esecuzione del programma. Si noti che quando una variabile compare con lo stesso nome in due blocchi diversi, viene mostrato il valore della variabile visibile nella parte di programma in esecuzione. Se si desidera vedere contemporaneamente le due variabili, si può usare la notazione **nomeblocco.variabile**, per distinguerle.

In questa fase è utile posizionare le finestre in modo da avere sempre sotto gli occhi la finestra **'watches'**, e quella del programma da controllare. Per ridimensionare una finestra si usa il comando **Size-move** del menù **Windows**. In tal modo muovendo il mouse o usando le frecce la finestra attiva si sposta e si può organizzare lo schermo a proprio piacimento.

3.4.4 Punti di arresto

L'esecuzione istruzione per istruzione risulta impraticabile non appena le dimensioni dei programmi e dei dati diventano significative. Si può per questo eseguire il comando **go to cursor** del menù **trace**, che provoca l'esecuzione di tutte le istruzioni da quella corrente fino al punto in cui è posizionato il cursore. Nella finestra **watches** compaiono i valori delle variabili aggiornate fino all'ultima istruzione eseguita.

Si può infine richiedere l'arresto dell'esecuzione del programma solo in determinati punti usando i comandi **Toggle breakpoint** o **Breakpoint** del menù **Debug**.

TOGGLE BREAKPOINT attiva e disattiva punti di arresto incondizionato, mentre **BREAKPOINT** permette di definire e controllare i punti di arresto. All'esecuzione di tale comando si apre una finestra in cui c'è l'elenco dei punti di arresto, la condizione ed il passaggio a cui tale punto di arresto si attiva. Se sono presenti dei punti di arresto incondizionati, cioè inseriti col comando **toggle breakpoint** non c'è né la condizione che li determina né il passaggio in cui vengono attivati. Se si vuole inserire un punto di arresto, si deve scegliere l'opzione **Edit** con **<alt>-e**: si apre così una finestra di dialogo in cui si può inserire:

CONDITION condizione,

PASS COUNT numero di passaggio prima dell'attivazione,

FILE NAME nome del file,

LINE NUMBER numero della riga del punto d'arresto.

3.5 Uso di file di dati esterni

Quando l'esecuzione di un programma richiede l'input di (o produce in output) una quantità di dati notevole, risulta conveniente memorizzare tali dati in un file, creato con l'editor se utilizzato per l'input, o creato dal programma e ispezionabile con l'editor, se usato per l'output. A questo scopo occorre procedere come segue.

Si dichiara una variabile di tipo file nel programma. Per esempio:

```
var F: text;
```

Si collega il file esterno alla variabile del programma con il comando **assign**.

```
assign(F,'miofile');
rewrite(F); {genera ed apre un file in scrittura}
```

Dopo questa istruzione tutte le operazioni su F saranno eseguite sul file in memoria di massa che si chiama miofile. Per le modalità di denominazione dei file si rimanda alla guida del sistema MS/DOS.

Si noti che il secondo argomento di `assign` può essere una variabile di tipo stringa, cui può essere assegnato un valore tramite un'istruzione di lettura (vedi esempio più avanti).

Si specifica il file in tutte le operazioni ad esso relative. Per esempio:

```
writeln(F, 'prova');
readln(F,a);
write(F, 'prova');
read(F,a);
```

Tutte queste istruzioni hanno il significato usuale ma sono eseguite sul file collegato alla variabile `F`. Si Chiude il file.

```
Close(F);
```

Ricordiamo che per aprire un file in lettura si usa:

```
reset(F);
```

Esempio di programma che memorizza una matrice su file esterno:

```
program memorizzamatricesufile;
var somma,i,j,numero,n,riga,colonna:integer;
    nomefile: string[8];
    F: text;
begin
    writeln('scrivi il nome del file su cui vuoi memorizzare');
    readln(nomefile);
    assign(F,nomefile);
    rewrite(F);
    writeln('ordine della matrice');
    writeln('righe: '); readln(riga);
    writeln('colonne: '); readln(colonna);
    writeln(F,riga);
    writeln(F,colonna);
    for i:=1 to riga do
    begin
        for j:=1 to colonna do
        begin
            writeln('scrivi il termine',i,',',j);
            read(numero);
            write(F,numero:4);
        end;
        writeln(F)
    end;
    close(F);
    readln
end.
```

3.6 I comandi dei menù

3.6.1 Menù File

OPEN Questo comando introduce ad una finestra di dialogo. In essa troviamo una casella, `NAME`, in cui va inserito il nome del file da caricare. Premendo il tasto freccia in giù compare una finestra in cui ci sono tutti i file che sono stati

aperti dall'inizio della sessione di lavoro, e che quindi possono essere facilmente richiamati. Sotto la casella NAME c'è la finestra FILES. Essa contiene il nome dei file nel direttorio corrente e che quindi è possibile caricare. Per entrare nella finestra si deve premere <alt>-f, e poi spostarsi sul file che interessa attraverso le frecce. Premendo invio si apre il file selezionato. Le operazioni possibili in questa finestra di dialogo sono: OPEN apre una nuova finestra di edit e mette il file selezionato in questa finestra. REPLACE invece di aprire una nuova finestra sostituisce il file nella finestra attiva con il file selezionato.

NEW Apre una nuova finestra di edit con il nome fittizio nomamexx.pas (xx è un numero da 00 a 99) ed automaticamente rende la finestra attiva.

SAVE Salva il file che è nella finestra attiva su disco. Se il file ha il nome fittizio nonamexx.pas allora viene automaticamente aperta la finestra (SAVE AS FILE).

SAVE AS FILE Non appena viene aperto questo sottomenù, il cursore si trova nella input-box ovvero la casella in cui si deve inserire il nome con il quale si vuole registrare il programma. Al di sotto di questa casella ho una finestra in cui c'è la lista dei file presenti nel direttorio corrente. Ancora al di sotto ho un' ultima finestra (blu) nella quale vengono date informazioni sui file già esistenti nel direttorio corrente (ad es. nome, data di esecuzione, memoria occupata).

GET INFO Mostra informazioni sul file nella finestra attiva. In essa non è possibile effettuare modifiche.

SAVE ALL Salva tutti i file nelle finestre aperte.

CHANGE DIR Il comando change dir conduce ad un sottomenù nel quale è possibile cambiare il direttorio corrente.

PRINT Stampa il contenuto della finestra attiva.

DOS SHELL Questo programma permette di abbandonare temporaneamente il TurboPascal ed eseguire comandi MS-DOS e programmi. Digitando EXIT si rientra nell'ambiente TurboPascal.

3.6.2 Menù Edit

RESTORE LINE Serve una volta modificata una linea a tornare allo stato precedente.

SHOW CLIPBOARD Apre la finestra clipboard, contiene ogni testo che si porta in essa attraverso i comandi cut e copy. I testi vengono inseriti uno di seguito all' altro. Il testo che compare evidenziato è quello che viene usato dal TurboPascal attraverso il comando paste.

CUT Si può usare solo dopo avere selezionato un testo. Il testo si seleziona in questo modo: si posiziona il cursore all'inizio del testo da selezionare, si preme Ctrl-k poi b, si posiziona il cursore alla fine del testo e si preme Ctrl-k poi k. Un modo molto più comodo è selezionare il testo premendo la freccia verso l'alto (Shift) e contemporaneamente spostare il cursore con le frecce sul testo da selezionare. Il comando cut (taglia) cancella il testo selezionato nella finestra attiva e lo colloca nella clipboard.

COPY E' simile al CUT: come la cut fa una copia del testo selezionato sulla clipboard tuttavia a differenza del cut tale comando lascia intatto il testo selezionato. E' possibile copiare testi dalle finestre help: si seleziona il testo nei modi precedentemente descritti e poi premendo <ctrl>-Ins si copia il testo nella clipboard.

PASTE Trasferisce il testo selezionato nella clipboard sulla finestra di edit attiva nella posizione indicata dal cursore. Si può ripetere questa operazione tutte le volte che si desidera.

COPY EXAMPLE I quadri di help contengono degli esempi di particolari procedure o funzioni che è possibile copiare nella clipboard e quindi trasferire nelle finestre di edit attraverso il comando paste.

CLEAR Cancella il testo selezionato; è una funzione irreversibile.

3.6.3 Menù Search

TEXT TO FIND Nella riga text to find va inserita la stringa da ricercare. Per iniziare la ricerca va scelto OK altrimenti va scelto cancel per tornare alla finestra attiva. Se si vuole inserire una stringa che si è già ricercato si preme la freccia verso il basso e viene mostrata l'history list, cioè la lista di tutte le ricerche fatte da inizio sessione di lavoro. Attraverso le frecce è possibile selezionare la stringa desiderata, premendo invio si avvia la ricerca. Se non si vuole ricercare una stringa già cercata basta digitarla e premere invio. Va notato che inizialmente nella riga del text to find compare la parola su cui si trova il cursore nella finestra di edit attiva. Inoltre vi sono le seguenti opzioni (OPTIONS):

Case sensitive: selezionandola il t.p. fa differenza tra maiuscole e minuscole

Whole word only: cerca solo le parole intere (e non sotto-stringhe)

Regular expression: riconosce i caratteri [] \ * . \$ ^

SCOPE Global: ci si riferisce al testo globale oppure **SCOPE Selected text:** ci si riferisce al testo selezionato

DIRECTION Forward: la direzione della ricerca va dall'origine alla fine oppure **DIRECTION Backward:** la direzione della ricerca è opposta.

ORIGIN From cursor: La ricerca inizia da dove è il cursore fino alla fine oppure **ORIGIN Entire:** scope La ricerca copre l'intero testo.

SEARCH AGAIN Ripete l'ultimo comando find o replace eseguito.

REPLACE Mostra una finestra di dialogo che consente di introdurre un testo da ricercare ed il testo da sostituirgli. Ha tutte le opzioni del comando text to find ,in più c'è un'opzione prompt on replace la quale se attivata fa sì che T.P. prima di operare la sostituzione chieda un'ulteriore conferma. Si può inoltre sfruttare la funzionalità change all che consente di cambiare tutte le corrispondenze trovate (e non solo la prima).

GO TO LINE NUMBER Sposta il cursore sulla linea di cui si è indicato il numero.

FIND PROCEDURE Attivo soltanto durante una fase di debugging, apre una finestra di dialogo in cui si può inserire il nome della funzione o della procedura da ricercare.

3.6.4 Menù Run

TRACE INTO Esegue il programma una riga per volta.

GO TO CURSOR L'opzione go to cursor serve per far eseguire al t.p. tutte le istruzioni dal begin fino al punto in cui è posizionato il cursore.

STEP OVER Consente di eseguire in un unico passo la chiamata ad una procedura, o funzione, senza cioè seguire il flusso completo della procedura.

PROGRAM RESET Ferma l'attuale sessione di debug, rilascia la memoria allocata, chiude ogni file aperto che il programma stava usando.

PARAMETERS Permette di passare argomenti al programma da eseguire come se fossero scritti nella riga di comando DOS.

RUN Esegue il programma fino al successivo punto d'arresto, o fino alla fine se non ce ne sono.

3.6.5 Menù Debug

EVALUATE MODIFY Calcola il valore di una variabile, o di un'espressione, lo visualizza e permette all'utente di cambiarlo.

WATCHES Apre un menù per la gestione dei punti di osservazione, cioè delle variabili di cui si vuole controllare l'evoluzione.

Add watches Inserisce un punto di osservazione; attivando questo comando compare una finestra di dialogo in cui bisogna inserire l'identificatore del punto di osservazione e premere invio. Inizialmente nella finestra compare la parola su cui è posizionato il cursore; è anche disponibile un elenco delle watches inserite dall'inizio della sessione di lavoro (si attiva premendo la freccia verso il basso).

Delete watch Cancella un punto di osservazione

Edit watch Serve per modificare l'espressione attiva nella finestra Watch. Selezionando questo comando compare una finestra di dialogo in cui c'è l'espressione che si vuole modificare

Remove all watches Cancella tutte le espressioni di controllo

TOGGLE BREAKPOINT Attiva e disattiva punti di arresto incondizionato.

BREAKPOINT Permette di gestire i punti di arresto

3.6.6 Menù window

SIZE/MOVE [ctrl-F5] Usando le frecce si può cambiare la posizione della finestra attiva (la cui cornice diventa azzurra). Scelta la posizione si preme [invio].

ZOOM [F5] Consente di ingrandire al massimo la finestra attiva. Se essa è già stata zoomata tale comando la riporta allo stato precedente.

TILE Serve a vedere contemporaneamente tutte le finestre di editing aperte. Tutte le finestre sono grandi uguali e messe una accanto all'altra senza sovrapposizioni.

CASCADE E' il modulo attivo quando si entra in t.p.: le finestre sono accatastate ed è visibile solo la finestra attiva, delle altre si vede solo il nome e il numero.

NEXT Rende attiva la finestra successiva.

PREVIOUS Rende attiva la finestra precedente.

CLOSE Chiude la finestra attiva.

WATCH Apre la finestra watch e la rende attiva.

REGISTER Apre la finestra Register che mostra i registri della CPU.

OUTPUT Apre la finestra Output e la rende attiva.

USER SCREEN Consente di vedere l'output di un programma a schermo pieno.

LIST Serve per ottenere un elenco di tutte le finestre aperte.

CALL STACK Apre una finestra che mostra la pila di chiamate a procedure e funzioni con i relativi parametri dalla più recente alla più vecchia. In fondo c'è quindi program nomeprogramma. Ovviamente la finestra è visibile solo se si sono predisposti punti di arresto, o si sta eseguendo il programma in Trace-into.

Nota: per attivare queste funzioni usare la combinazione **alt<carattere in neretto>** dove il carattere in neretto è quello che si vede nella voce del comando nel menù.

3.7 Librerie di sottoprogrammi (unit)

Il TurboPascal possiede nella sua libreria alcune funzioni e procedure che l'utente può usare senza doverle riscrivere (ad es. sin,cos,ecc.). Esistono inoltre delle funzioni più specifiche contenute in librerie chiamate UNIT alle quali si può accedere solo eseguendo particolari istruzioni. Le unit predefinite sono: system, overlay, crt, dos, printer

3.7.1 Uso delle unit standard

L'uso è molto semplice basta dichiarare dopo l'intestazione del programma l'istruzione :

```
uses nome programma;
```

per esempio:

```
uses overlay;
```

a questo punto è possibile usare tutte le funzioni e procedure di overlay.

Segue una breve descrizione delle unit predefinite

system System contiene tutte le procedure e le funzioni standard del TurboPascal.

Per esempio

```
program pr2;
  {ogni programma o unit usa system, per cui non c'è bisogno di
  dichiarare uses system}
begin
  Mkdir(ParamStr(1)); {crea un sottodirettorio, che ha per nome}
                     {il parametro che si è inserito in }
                     {RUN-parameters}
```

```

    if IOResult <> 0 then {IOResult restituisce il risultato
                        {dell'ultima operazione di }
                        {input-output: restituisce un intero=0}
                        {se l'operazione ha avuto successo}
        WriteLn('Cannot create directory')
    else
        WriteLn('New directory created');
end.
```

Se si è inserita tra i parametri (RUN-parameters) una stringa, il programma crea un direttorio che ha per nome la stringa inserita come parametro.

dos Dos definisce numerose procedure e funzioni Pascal analoghe alle più frequenti chiamate DOS, come exec, get time, set time, disk size, ecc. es.

```

program prova;
uses Dos;
const
    days : array [0..6] of String[9] = ('Sunday','Monday',
    'Tuesday','Wednesday','Thursday','Friday', 'Saturday');
var
    y, m, d, dow : Word;
begin
    GetDate(y,m,d,dow);
    {questa funzione restituisce la data corrente}
    {(anno,mese,giorno,giornodella settimana)}
    WriteLn('Today is ', days[dow],',', ' ', m:0, '/', d:0, '/', y:0);
end.
```

overlay Overlay fornisce il supporto al potente sistema di overlay di TurboPascal. Quando un programma è troppo lungo e quindi non può stare completamente in memoria centrale il TurboPascal lo divide in moduli e carica man mano il modulo che serve in quella fase del programma. Overlay consente di fare ciò in maniera più elastica.

crt Crt fornisce un insieme di dichiarazioni per l'input output. Utilizzando queste definizioni è possibile manipolare i testi sullo schermo(finestre , indirizzamento diretto del cursore,gestione del colore nei testi). es.

```

program ex;
uses Crt;
begin
    TextBackground(LightGray); {lo sfondo diventa grigio chiaro}
    TextColor(black); {il colore del testo diventa nero}
    clrscr; {pulisce lo schermo}
    writeln('prova');
end.

program pr2;
uses Crt;
begin
    repeat
        Write('Xx'); {riempie lo schermo di Xx}
        {finchè non si preme un tasto}
    until KeyPressed;
end.
```

printer Printer dichiara la variabile LST per i file di testo e la connette con un driver di dispositivo che consente di effettuare l'output con il Pascal standard tramite write e writeln.

graph La unit graph mette a disposizione un insieme di routine grafiche veloci e potenti che permette di sfruttare appieno le possibilità grafiche del proprio PC.

3.7.2 Costruire una unit

In Pascal è possibile raccogliere procedure o funzioni in una propria unit ed utilizzarle in un programma ogni volta che si vuole senza doverle ridichiarare e riimplementare.

Possiamo quindi definire una unit come una struttura che esporta all'esterno una serie di funzionalità attraverso le procedure e funzioni in essa contenute, nascondendo all'utente il modo in cui queste funzionalità vengono realizzate.

L'intestazione inizia con la parola unit seguita dal nome con cui si vuole chiamare la unit. L'elemento successivo è la parola interface, che indica l'inizio della sezione di interfaccia. L'interfaccia è una sezione di dominio pubblico cioè è accessibile da qualsiasi altra unit o programma che la usi: essa contiene gli elementi che ciascun programma può usare.

Nell'interfaccia si possono dichiarare variabili, procedure, funzioni e tipi di dati (la sequenza in cui vanno dichiarati è la stessa della dichiarazione nei programmi).

Le funzioni e procedure accessibili da un programma vengono dichiarate qui: solo dichiarate e il corpo principale va inserito nella sezione successiva all'interface, cioè la sezione implementation (la parte privata invisibile ai programmi).

La sezione implementation contiene quindi l'usuale implementazione delle procedure e funzioni dichiarate nell'interfaccia. Tutto ciò che è stato dichiarato precedentemente nella sezione di interfaccia è accessibile alla sezione di implementazione, cioè si possono usare in essa le variabili e tipi definiti nell'interfaccia. Si possono inserire altre dichiarazioni esclusive alle quali i programmi esterni non possono accedere. Le dichiarazioni delle procedure e funzioni presenti nell'interfaccia devono comparire anche nella sezione implementazione ovviamente la dichiarazione deve essere la stessa oppure in forma abbreviata, omettendo parametri e tipo della funzione. Ad esempio:

INTERFACE

.....

procedure esempio(var a:integer;b,c:char);

.....

IMPLEMENTATION

.....


```

procedure esempio
begin
.....
.....
end;

```

Infine può esserci una parte dedicata all'inizializzazione. Essa inizia con begin e termina con end. come la parte esecutiva di un programma. Questa parte può servire ad inizializzare le strutture dati utilizzate dalla unit o dal programma che la utilizza. Ad esempio:

```

UNIT un_mat;

INTERFACE

  uses crt;
  const M=3;      {DIMENSIONE DELLE MATRICI}
  type matrice=array [1..M,1..M] of integer;
  var mat,mat3:matrice;
      i,j:integer;
      t,t1:text;

  procedure scrivimatsufile(mat3:matrice;var t1:text);
  procedure leggimatricedafile(var t:text;var mat:matrice);
  procedure leggimat(var mat:matrice); { Inizializza la matrice }
  procedure visualizzamat(mat:matrice); { Visualizza la matrice }

IMPLEMENTATION

  procedure leggimat(var mat:matrice); {Inizializza la matrice}
    var rig,col:integer;
  begin
    for rig:=1 to M do
      begin
        writeln;
        for col:=1 to M do
          begin
            writeln ('Inserisci valore di riga ',rig,' e colonna',col);
            readln (mat[rig,col])
          end
        end
      end
    end;

  procedure visualizzamat(mat:matrice); { Visualizza la matrice}
    var rig,col:integer;
  begin
    for rig:=1 to M do
      begin

```

```

        writeln;
        for col:=1 to M do
            write (mat[rig,col]:7);
        end
    end;

procedure scrivimatsufile(mat3:matrice;var t1:text);
var i,j:integer;uscita:string[8];
begin
    writeln('dammi il nome del file su cui vuoi memorizzare la matrice');
    readln(uscita);
    assign(t1,uscita);
    rewrite(t1);
    for i:=1 to M do
        begin
            for j:=1 to M do write(t1,mat3[i,j]:7);
                writeln(t1)
            end;
        close(t1);
    end;

procedure leggimatricedafile(var t:text;var mat:matrice);
var i,j:integer;dati:string[8];
begin
    writeln('dammi il nome del file su cui sono contenuti i dati');
    readln(dati);
    assign(t,dati);
    reset(t);
    for i:=1 to M do
        for j:=1 to M do
            read(t,mat[i,j]);
        close(t);
    end;

begin
    textmode(co40);
    window(1,1,100,160);
    textbackground(blue);
    textcolor(white);
    clrscr;
end.

```

In questo esempio nella sezione di inizializzazione compaiono le chiamate a funzioni di libreria che servono per pulire lo schermo, rendere blu lo sfondo, il testo bianco ed i caratteri più grandi.

3.7.3 Compilazione di una unit

Per poter usare la unit questa deve essere stata precedentemente compilata e memorizzata in linguaggio macchina e non nel sorgente Pascal. Cioè non basta scrivere il testo di una unit o richiamarla da disco per poi poterla usare.

La unit deve essere memorizzata in linguaggio macchina.

Per far ciò è necessario accedere al menù compile,(alt+c) spostarsi su Destination. Se accanto c'è la scritta memory premere invio. In tal modo la finestra

si chiude automaticamente e Destination si posiziona su disk. Si riapre il menù compile, si seleziona compile e se non vengono rilevati errori e la compilazione va buon fine si genera un file nomeunit.tpu . A questo punto la unit può essere utilizzata.

3.7.4 Funzionalita' Della Unit

La unit dell'esempio precedente consente ad un programma esterno di usare il tipo matrice le variabili t, t1 di tipo testo, mat, mat3 di tipo matrice.

Si possono usare le procedure:

`leggimat(a)` chiede all'utente di inserire da tastiera una matrice che memorizza nella variabile a

`visualizzamat(a)` visualizza la matrice contenuta nella variabile a.

`scrivimatsufila(a)` analoga a `visualizzamat`, ma memorizza la matrice a la su un file.

`leggimatdafila(a)` analoga `leggimat`, ma legge la matrice da file.

3.7.5 Uso della unit

L'uso di una unit è molto semplice, basta dichiarare `uses <nomeunit>` dopo l'intestazione del programma. Ad esempio:

```
program prova;
uses un_mat;
begin
  writeln('dammi matrice da salvare');
  leggimat(m1);
  scrivimatsufila(m1,t5);
  leggimatricedafila(t3,m2);
  visualizzamat(m1);
  writeln;
  visualizzamat(m2);
end.
```

Se la unit in linguaggio macchina (un file **nomeunit.tpu**) è nel direttorio corrente (nello stesso direttorio del programma che la usa) non ci sono problemi, altrimenti bisogna aprire il menù option-directories, ed indicare nella riga unit directories il direttorio in cui è contenuta la unit che si desidera utilizzare.

4 Free Pascal

Il compilatore freepascal è un ottimo compilatore freeware multiplatforma , infatti il FP è disponibile per diversi sistemi operativi tra cui Unix/Linux e Dos/Windows.

4.1 Installazione

Partiamo dall'installazione del compilatore FreePascal: andate sul sito www.freepascal.org alla sezione download e scegliete il mirror da dove scaricare i file.

A questo punto dovete decidere quali file scaricare; i file sono raggruppati per sistema operativo quindi scegliete il sistema operativo sul quale deve girare il compilatore.

Se avete scelto DOS o WIN32 potete scaricare un unico pacchetto di circa 22 MByte che contiene praticamente tutto, file di base, opzionali e codice sorgente di esempio. Cmq a voi la scelta dei pacchetti da scaricare: per Dos e Win32 i pacchetti

di base richiedono un download di circa 1,8 MByte. Personalmente io ho scaricato il pacchetto full contenente tutto, ma non c'è alcun problema dato che la procedura di installazione è la stessa.

Se volete usare il freepascal su un sistema Linux/Unix dovete scaricare o il pacchetto completo di circa 12 MByte, contenente uno script di installazione standard o i pacchetti per RedHat o Debian. Cmq se usate questo sistema operativo non avete bisogno che vi spieghi come installare il compilatore.

Vediamo in dettaglio come effettuare l'installazione sotto DOS/WIN32:

Dopo avere scaricato i file eseguite il file `install.exe`.

Nella sezione GENERAL dovete specificare il percorso della directory in cui installare il compilatore e selezionare la casella `Create ppc386.cfg` per fare in modo che venga creato automaticamente il file di configurazione.

Adesso dovete selezionare il software da installare; a seconda dei file che avete scaricato ci sono diverse schermate di selezione: DOS, WIN32, COMMON e SOURCES.

La schermata DOS contiene tutti i file per il compilatore DOS, i pacchetti minimi richiesti sono: Basic System e GNU linker e assembler quindi questi devono essere selezionati necessariamente; inoltre si possono installare delle utility aggiuntive come: l'IDE cioè l'ambiente di sviluppo integrato, che è molto simile agli ambienti Borland per DOS; oppure è disponibile il GNU Debugger... e molto altro. Partite sempre dal sito di riferimento [7].

La sezione COMMON contiene la varia documentazione, invece SOURCES contiene il codice sorgente del compilatore e di altri componenti.

Dopo avere selezionato quello che interessa premete il tasto CONTINUE, se tutto è andato bene il programma termina dicendovi di aggiungere il percorso del compilatore al vostro PATH.

Aperte l'`autoexec.bat` e aggiungete:

```
SET PATH=c:\pp\bin\go32v2\%PATH%
```

Naturalmente dovete mettere il vostro percorso, questo è il percorso nel mio PC. A questo punto il compilatore è installato dobbiamo solo vedere come usarlo.

4.2 Uso del compilatore

Ci sarebbero da scrivere pagine e pagine, ma lo spazio non può essere illimitato. Cominciamo col dire che l'interfaccia grafica è veramente simile a quelle del Turbo Pascal, quindi potete consultare la sezione precedente per avere informazioni riguardanti anche il Free Pascal. Per quando riguarda gli IDE che abbiamo citato precedentemente, rimando ai loro manuali (ovviamente in inglese!).

Adesso che abbiamo installato il compilatore FreePascal vediamo come usarlo; non mi soffermerò sull'uso dell'IDE, ma solo sull'uso del compilatore con le sue relative opzioni.

Il nome del compilatore è `ppc386.exe` sotto DOS o WINDOWS, e `ppc386` sotto LINUX; l'uso elementare del compilatore a riga di comando è banale, in quanto per rendere eseguibile un programma pascal, contenuto per esempio nel file `prova.pas`, basta lanciare il comando:

```
ppc386.exe prova.pas (DOS)
ppc386 prova.pas (LINUX)
```

Se il programma contenuto in `prova.pas` è corretto, la compilazione va a buon fine e viene creato il file eseguibile.

Se invece il programma contiene degli errori la compilazione si non crea il file eseguibile, ma vi vengono segnalati gli errori con un messaggio del tipo:

NomeFile.pas(r,c) ERROR : descrizione

Dove *r* e *c* indicano la riga e la colonna in cui si trova l'errore all'interno del file; descrizione invece specifica qual è il tipo di errore riscontrato, naturalmente in inglese.

Quindi non appena si riscontrano degli errori bisogna correggerli, facendo riferimento ai messaggi del compilatore e ricompilare il programma al fine di renderlo eseguibile.

4.3 Opzioni del compilatore

Analizziamo ora alcune opzioni messe a disposizione dal compilatore.

Prima di andare ad analizzare le opzioni vediamo come si specificano le stesse al compilatore:

```
ppc386.exe [opzioni] prova.pas (DOS)
ppc386 [opzioni] prova.pas (LINUX)
```

Quindi bisogna inserire l'opzione tra il nome del compilatore e il nome del file di input.

Adesso andiamo nel dettaglio e vediamo alcuni di queste opzioni:

- a	Genera il listato in asm del nostro programma
- g	Genera le informazioni per il debug; senza questa opzione non è possibile effettuare il debugging.
- iD - iV - iSO	Queste opzioni non influiscono sulla compilazione, ma danno delle informazioni sul compilatore. Le opzioni indicati forniscono rispettivamente: data, versione e sistema operatore del compilatore.
- o nome	Setta il nome del file eseguibile a quello specificato.
- Og - OG - O1 - O2 - O3	Queste opzioni sono di ottimizzazione del codice; nell'ordine: generazione di codice ridotto (dimensione contenute al massimo); generazione di codice veloce (tempo di esecuzione più rapido), questa opzione è quella di default cioè quella che viene eseguita in assenza di specifiche diverse; le ultime 3 invece generano i tre livelli di ottimizzazione 1, 2 e 3.
- Fu	path Aggiunge la directory specificata alla lista delle directory in cui ricercare le unit.
- TGO32V2 - TLINUX - TOS2 - TWin32	Queste opzioni specificano il sistema operativo; nell'ordine: DOS, LINUX, OS/2 e WINDOWS

Queste sono solamente alcune delle moltissime opzioni disponibili per il compilatore FreePascal; io ho inserito queste perchè le ritengo quelle più utili e che potrebbero servire. Nella maggior parte dei casi non c'è bisogno di utilizzare alcuna di queste opzioni, ma basta utilizzare la forma semplice descritta all'inizio.

Riferimenti bibliografici

- [1] Enrico Centenaro. *Appunti di Programmazione Strutturata*. Scrittura privata, 1997-2000.
- [2] Vari. *Appunti sul Pascal*. Dispense, 1987.
- [3] Niklaus Wirth. *Algoritmi + Strutture Dati = Programmi*. Casa Editrice... anno...
- [4] D. E. Knuth *The TeXbook*. Addison-Wesley. 1984, ISBN 0-201-13447-0.

- [5] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison-Wesley, 1994.
- [6] Borland, <http://www.borland.com>.
- [7] Open Source, <http://www.freepascal.org>.