Vornicescu Silviu

Programarea în Turbo Pascal

Introducere în Turbo Pascal

Limbajul Pascal a fost elaborat în **1972** de către **Nicklaus Wirth** de la Universitatea din Zurich în scopul unei mai bune învăţări a programării. Deşi la început el a fost cunoscut doar de mediul universitar, apoi el a cunoscut un succes imens.

Structura programelor Pascal

1. Orice program începe printr-un cuvînt numit ***PROGRAM*** care este urmat de numele propriu-zis al programului, adică de un nume ales de utilizator şi de semnul **;** (punct şi virgulă)
2. Orice program conţine cel puţin odată cuvintele cu un înţeles special ***BEGIN*** şi ***END***
3. Orice program se termină prin punct
4. Orice cuvînt al programului poate fi scris cu litere mari sau mici, deoarece nu are importanţă
5. În versiunea ***Turbo,*** prima linie poate lipsi, deşi nu se recomandă acest lucru, din raţiuni de ordine
6. Plasarea cuvintelor pe linie şi numărul de spaţii dintre ele sînt la alegerea programatorului. Se poate scrie tot programul pe o singură linie, însă este bine ca programul să fie scris în aşa fel încît să fie uşor de înţeles.

Un program scris în Pascal, oricît de complex ar fi, are următoarea structură :

PROGRAM nume ;

- definiţii de constante ;
- definiţii de tipuri ;
- declaraţii de variabile ;
- declaraţii de proceduri şi funcţii ;
- **BEGIN**
- Instrucţiuni
- **END**

Nu este obligatoriu ca într-un program să figureze toate acestea, dar dacă ele figurează, trebuie să apară în această ordine.

Orice program cuprinde două părţi esenţiale :

- descrierea instrucţiunilor (a acţiunilor) şi
- descrierea datelor care sînt prelucrate de instrucţiuni

Un program Turbo Pascal este structurat astfel :

a) **un antet de program**, care conţine cuvîntul rezervat **PROGRAM** urmat de numele programului
b) **o declaraţie**, introdusă în cuvîntul rezervat **USES**, a numelor **unit**-urilor, care se vor folosi într-un program
c) o parte obligatorie formată dintr-un bloc, urmat de caracterul punct

Componentele **a)** şi **b)** , de mai sus, pot să lipsească dintr-un program

Un **UNIT** este o colecţie de declaraţii de constante, tipuri, variabile şi subprograme în cod obiect (adică compilate separat), care se utilizează în program prin specificarea numelui său în clauza **Uses**. Folosirea unit-urilor permite scrierea unor programe mari care depăşesc 64 K. Dimensiunea unui program, ca şi ceea a unui unit, nu poate depăşi 64 K. Numărul unit-urilor utilizate de un program nu este însă limitat şi de aceea programul poate atinge dimensiuni care depind de capacitatea memoriei interne. Utilizatorii pot să-şi creeze unit-uri proprii dar pot utiliza şi cele 8 unit-uri standard oferite de mediul Turbo Pascal : **SYSTEM, DOS, GRAPH, CRT, OVERLAY, PRINTER, GRAPH3, TURBO3.** Unit-ul **SYSTEM** este încorporat în toate programele Pascal, fără să fie obligatorie declararea lui în **USES**, deoarece el cuprinde procedurile şi funcţiile cel mai des folosite. Folosirea unit-urilor, nu este posibilă în Pascal Standard.

Blocul, este elementul de bază al programului şi cuprinde o secţiune de declaraţii, care este opţională şi o secţiune obligatorie de program.

Elementele limbajului

Turbo Pascal foloseşte următorul subset al caracterelor ASCII **:**

- litere **: a – z, A – Z**
- cifre **: 0 – 9**
- cifre hexa **: 0 – 9; A – F; a – f**
- caractere speciale **: + - * / @ # $ % ^ & { }**
- caracterele ASCII 0 – 32

Cuvintele cheie (rezervate)

Aceste cuvinte pot fi scrise atît cu litere majuscule, cît şi cu minuscule sau pot fi chiar şi amestecate. Cuvintele rezervate sînt cuvinte care au un înţeles special, ele neputînd fi redefinite de utilizator.

And	End	Label	Repeat
Asm	Exports	Library	Set
Array	File	Mod	Shl
Begin	For	Nil	Shr
Case	Function	Not	String
Const	Goto	Object	Then
Constructor	If	Of	Until
Destructor	Implementation	Or	Uses
Div	In	Packed	Var
Do	Inherited	Procedure	While
Downto	Inline	Program	With
Else	Interface	Record	Xor

Directive standard

Spre deosebire de cuvintele rezervate, *directivele standard* pot fi redefinite de programator, aceasta însă nu este recomandat.

Absolute	Far	Near	Virtual
Assembler	Forward	Private	
Export	Index	Public	
External	Interrupt	Resident	

Identificatorii

Sînt simboluri care desemnează ***variabile, proceduri, funcţii, unit-uri, constante, tipuri, programe şi cîmpuri de articole.*** Un identificator începe cu o literă, care poate fi urmată de orice combinaţie de litere, cifre sau caracterul de subliniere (_ , ASCII $5F). Lungimea identificatorilor nu este limitată, dar sînt semnificative doar primele 63 de caractere.

În cazul programelor ce folosesc unit-uri, pot exista instanţe cu acelaşi identificator. Pentru a putea selecta o instanţă specifică, identificatorul ales, trebuie calificat. Calificarea se realizează printr-o construcţie de forma **:**

Nume_unit.identificator

Adică numele identificatorului trebuie să fie precedat de numele unit-ului de origine, urmat de un punct.

Exemplu **: dos.exec**

O altă categorie specială de identificatori sînt ***tipurile predefinite*** ale limbajului. Aceste sînt următoarele **:**

Boolean	Comp	Longbool	Pointer	Text
Byte	Double	Longint	Real	Word
Bytebool	Extended	Openstring	Shortint	Wordbool
Char	Integer	Pchar	Single	

O altă categorie specială de identificatori o reprezintă, directivele ***name, index, resident*** (folosite de biblioteci de legare dinamică **.DLL**), precum şi parametrul ***Self***.

Atît tipurile predefinite, cît şi directivele precedente pot fi redefinite, dar această redefinire nu este recomandată.

Etichetele

Sînt specificate sub formă de numere întegi din domeniul **0…9999** sau de identificatori. Etichetele sînt utilizate la instrucţiunile de salt necondiţionat ***goto***. Dacă o instrucţiune este precedată de o etichetă, este permisă folosirea ei prin instrucţiunea ***goto***. O etichetă marchează numai o linie. Eticheta este separată de instrucţiunea corespunzătoare prin caracterul două puncte (**:**). Toate etichetele utilizate trebuie declarate în secţiunea ***label***.

Exemplu de etichetă :

```
Label 1234;
   {…}
begin
   {…}
1234: y:=sin(x)+1.0;
   {…}
   goto 1234;
   {…}
end.
```

Constantele, numerele

Constantele, desemnează valori întregi sau reale. În mod obişnuit se foloseşte notaţia zecimală, dar este permisă şi folosirea notaţiei hexazecimale, în cazul numerelor întregi. O constantă hexazecimală, foloseşte semnul dolarului (**$**) ca prefix.

Numerele reale se scriu sub forma **:**

partea_întreagă.partea_zecimală

adică în locul virgulei zecimale, se foloseşte punctul zecimal. Opţional se poate adăuga şi un exponent, la care litera **E** sau **e** este urmată de un exponent cu sau fără semn. Dacă exponentul este prezent, numărul este înmulţit cu 10 la puterea exponentului. Numerele hexa trebuie să fie cuprinse între **$00000000** şi **$FFFFFFFF** , iar numerele întregi trebuie să fie cuprinse între **–2,147,483,648** şi **2,147,483,647**.

Operatorii

Limbajul Turbo Pascal 7.0 are nouă tipuri de operatori **:**

1) asignare,	4) relaţionali	7) mulţime
2) aritmetici,	5) logici	8) asamblare
3) lucru cu biţi	6) adresare	9) şir

Unii operatori sînt binari, solicitînd doi operanzi, restul sînt unari, solicitînd un operator.

Precedenţa (priorităţile) operatorilor

În expresii complexe, regulile de precedenţă (prioritate), clarifică ordinea în care se vor executa operaţiile. Tabelul următor prezintă priorităţile :

Operator	Prioritate
@ not	1
* / div mod and shl shr	2
+ - or xor	3
= <> < > <= >= in	4

Operatorii care au aceiaşi prioritate sînt executaţi în mod normal de la stînga la dreapta. Expresiile cu paranteze sînt evaluate mai întîi, independent de operatorii precedenţi sau de succesori.

1) Operatorul de asignare

Una din cele mai frecvente operaţii în cadrul programelor este asignarea, adică operaţia de atribuire a unei valori, unei variabile. Simbolul de asignare este format din caracterul două puncte (**:**) urmat de caracterul egal (**=**).
Exemplu : **A:=3;**

2) Operatorii aritmetici

*****	Multiplicare, înmulţire
Div	Împărţire întreagă
/	Împărţire cu rest
Mod	Modul
+	Adunare
-	Scădere

3) Operatori pentru biţi

SHL	(shift left)
Shr	(shift right)
And	
Or	
Xor	
Not	

4) Operatori relaţionali

Operatorii relaţionali permit compararea a două valori şi returnează rezultatul ***boolean*** , **TRUE** sau **FALSE**.

>	mai mare
>=	mai mare sau egal cu
<	mai mic
<=	mai mic sau egal cu
=	egal
<>	diferit
in	apartenenţa

5) Operatorii logici

Limbajul Turbo Pascal are patru operatori logici **:**

and
xor
or
not

care sînt similari, dar nu identici cu operatorii pentru biţi.

6) Operatorii de adresare

Limbajul Turbo Pascal foloseşte doi operatori de adresare **:**

- adresa lui @, care returnează adresa unei variabile
- (**^**) – operatorul de indirectare

7) Operatorii pentru mulţimi

+	reuniunea
-	diferenţa
*****	intersecţia

8) Operatorii limbaj de asamblare

&	redefinire identificator
(…)	subexpresie
[…]	referinţă memorie
HIGH	întoarce octetul cel mai semnificativ
LOW	întoarce octetul cel mai puţin semnificativ
+	plus unar
-	minus unar
:	redefinire segment
OFFSET	întoarce partea offset
SEG	întoarce partea segmentului
TYPE	întoarce tipul, în octeţi
PTR	
*****	multiplicare
/	diviziune
MOD	modul
SHL	shift logic stîng
SHR	shift logic drept
+	adunare binară
-	scădere binară
NOT	negare
AND	intersecţie
OR	reuniune
XOR	reuniune exclusivă

9) Operatorul pentru şiruri

Singurul operator pentru şiruri, este operatorul plus (**+**), care permite concatenarea (alipirea) a două şiruri.

Separarea instrucţiunilor

Este realizată prin caracterul punct şi virgulă (**;**). Acest caracter este destinat separării şi nu terminării unei instrucţiuni. Prezenţa lui **nu** este obligatorie după fiecare instrucţiune iar în faţa instrucţiunii **End**, poate să şi lipsească, iar în faţa cuvîntului cheie **Else**, acest caracter este interzis.

Şiruri de caractere

Un şir de caractere este o secvenţă de caractere încadrate între două apostroafe. Dacă se doreşte ca şirul însuşi să conţină caracterul apostrof, acest caracter trebuie să fie dublat. Şirul poate să conţină zero sau mai multe caractere din setul extin de caractere ASCII. Un şir nu poate fi despărţit în mai multe linii. Un şir de caractere, care nu conţine nici un caracter, se numeşte ***şir vid (nul)***. Limbajul Turbo Pascal permite introducerea caracterelor în şiruri şi în felul următor **:** semnul **#** urmat de un număr natural din domeniul **0…255** desemnează un caracter din **setul ASCII extins**. Între **#** şi numărul natural, nu trebuie să existe nici un separator. Dacă mai multe caractere astfel scrise, sînt părţi ale unui şir, atunci nici aici nu sînt folosiţi separatori.

Exemple de şiruri de caractere :

‘Turbo’
#13#10#13
#7#7’Atenţie’#7#7
‘You’’ll see’
‘Linia 1’#13’Linia 2’

Comentarii

Un comentariu este o secvenţă de caractere inserate în program cu scopul facilitării înţelegerii acţiunilor sale. Comentariile sînt **construcţii ignorate de compilator** şi servesc la precizarea observaţiilor programatorului, privitor la o secţiune din program. Un comentariu începe cu caracterul **{** sau cu perechea de caractere **(*** şi se încheie cu caracterul **}** sau cu perechea ***)**. Un comentariu care conţine semnul dolar ($), imediat după deschiderea sa, reprezintă o directivă de compilare. Comentariile pot să conţină orice secvenţă de caractere exceptînd pe cele menţionate mai sus.

Exemple de comentarii :

{ Acesta este un comentariu **}**
(* Acesta este un program ***)**

Directive de compilare

Directivele de compilare, sînt comentarii cu o sintaxă specială, şi pot fi utilizate peste tot unde sînt admise comentarii. Directivele de compilare încep cu caracterele **{$** sau **(*$**, care sînt urmate de numele directivei şi se termină cu caracterul **}** sau ***)**. Există trei categorii de directive **:**

- **Directivele tip comutator**, prin care pot fi activate diferite funcţii ale compilatorului. Numele directivei este urmat de caracterul **+** pentru cuplare sau de caracterul **–** pentru decuplare.
- **Directivele tip parametru**, specifică anumiţi parametri care afectează modul de compilare.
- **Directivele condiţionale**, controlează compilarea condiţională a anumitor părţi ale textului sursă.

Directivele de compilare tip comutator

Directivele de compilare tip comutator furnizează anumite informaţii necesare compilatorului. Ele pot fi cuplate sau decuplate. De regulă, după caracterul "$", apare o literă şi semnul "**+**" sau "**-**", de exemplu **{$F+}**, **{$R-}**. Ele se referă la anumiţi comutatori (valori logice). Directivele de compilare pot fi **:**

- locale
- globale

Directivele locale pot să apară oriunde în program sau în unit, pentru că ele afectează doar o parte a compilării.

Directivele globale trebuie să fie plasate înaintea părţii de declarare a programului sau a unit-ului care se compilează, pentru că ele afectează întreaga compilare.

Se permite gruparea mai multor directive de compilare, care trebuie separate cu virgulă.

Exemplu :

{$F+, R+, E-, D-}

Directivă	Descriere	Valoare implicită	Efect
{$A+} {$A-}	**Aliniere date**	**Cuplată [X]**	**Global**
{$B+} {$B-}	**Evaluare booleană**	**Decuplată [ ]**	**Local**
{$D+} {$D-}	**Informaţii de depanare**	**Cuplată [X]**	**Global**
{$E+}	**Emulare**	**Cuplată [X]**	**Global**

{$E-}			
{$F+} {$F-}	**Model de apel far**	**Decuplată [ ]**	**Local**
{$G+} {$G-}	**Generare de instrucţiuni 286**	**Decuplată [ ]**	**Local**
{$I+} {$I-}	**Testare operaţii intrare / ieşire**	**Cuplată [X]**	**Local**
{$L+} {$L-}	**Informaţii simboluri locale**	**Cuplată [X]**	**Local**
{$N+} {$N-}	**Prelucrare numerică**	**Decuplată [ ]**	**Global**
{$O+} {$O-}	**Structură de acoperire. Numai în mod real !**	**Decuplată [ ]**	**Global**
{$P+} {$P-}	**Parametru şir deschis de caractere**	**Decuplată [ ]**	**Global**
{$Q+} {$Q-}	**Testare depăşire numere întregi**	**Decuplată [ ]**	**Local**
{$R+} {$R-}	**Testare domeniu de valori**	**Decuplată [ ]**	**Local**
{$S+} {$S-}	**Testare depăşire stivă**	**Cuplată [X]**	**Local**
{$T+} {$T-}	**Operator de adresare @ cu tip**	**Decuplată [ ]**	**Local**
{$V+} {$V-}	**Test identitate tip la parametru / argument şir**	**Cuplată [X]**	**Local**
{$X+} {$X-}	**Sintaxa extinsă**	**Decuplată [ ]**	**Global**
{$Y+} {$Y-}	**Informaţii tip browser Numai în mod protejat !**	**Cuplată [X]**	**Global**

Directivele de compilare şi echivalenţele lor în meniul Option/Compiler

Directivă	Grupa	Echivalent
{$A+}** **{$A-}	**Generare de cod** **Generare de cod**	**[X] Word Align Data** **[ ] Word Align Data**
{$B+} {$B-}	**Sintaxa** **Sintaxa**	**[X] Complete Boolean eval.** **Boolean Evaluation. Short Circuit**

{$D+} {$D-}	Depanare Depanare	Debug Information.On Debug Information.Off
{$E+} {$E-}	Prelucrare numerică Prelucrare numerică	Emulation.On Emulation.Off
{$F+} {$F-}	Generare de cod Generare de cod	Force Far Calls.On Force Far Calls.Off
{$G+} {$G-}	Generare de cod Generare de cod	286 Code.On 286 Code.Off
{$I+} {$I-}	Erori de execuţie Erori de execuţie	I/O Checking.On I/O Checking.Off
{$L+} {$L-}	Depanare Depanare	Local Symbols.On Local Symbols.Off
{$N+} {$N-}	Prelucrare numerică Prelucrare numerică	80x87 Code.On 80x87 Code.Off
{$O+} {$O-}	Generare de cod Generare de cod	Overlay Code Generation.On (Numai în mod real) Overlay Code Generation.Off (Numai în mod real)
{$P+} {$P-}	Sintaxă Sintaxă	Open string parameters. Enabled Open string parameters. Disabled
{$Q+} {$Q-}	Erori de execuţie Erori de execuţie	Overflow Checking.On Overflow Checking.Off
{$R+} {$R-}	Erori de execuţie Erori de execuţie	Range Cecking.On Range Cecking.Off
{$S+} {$S-}	Erori de execuţie Erori de execuţie	Stack Cecking.On Stack Cecking.Off
{$T+} {$T-}	Sintaxă Sintaxă	Typed @ operator.On Typed @ operator.Off
{$V+} {$V-}	Sintaxă Sintaxă	Strict Var – String Checking Relaxed Var – String Checking
{$X+} {$X-}	Sintaxă Sintaxă	Extended Syntax.On Extended Syntax.On
{$Y+} {$Y-}	Depanare Depanare	Symbol reference information. On (numai în mod real) Symbol reference information. Off (numai în mod real)

Directive tip parametru

Directivele tip parametru specifică parametri care afectează modul de compilare a programului.

Directive parametru	Semnificaţie
$C Atribut	Atributul segmentului de cod (sub Windows şi în mod protejat)
$D Text	Descriere (sub Windows şi în mod protejat)
$I Numefişier	Includerea fişierului sursă, cu extensia **.PAS**
$G NumeUnit	Segmente de grup unit (sub Windows şi în mod protejat)
$L Numefişier	Includerea fişierului obiect, cu extensia **.OBJ**
$M DimensiuneStivă, HeapMinim, HeapMaxim	Dimensiunile de alocare a memoriei : stivă şi valori extreme pentru **heap**.
$O NumeUnit	Numele unit-ului de acoperire, cu extensia **.OVR** (numai în mod real)
$R NumeFişier	Fişier de resurse (sub Windows şi modul protejat)
$S DimSegment	Preferinţă dimensiune segment (sub Windows şi în mod protejat)

Între numele directivei şi parametrii trebuie plasat cel puţin un caracter blanc.

Exemplu :

{I TYPES.INC}
{$O XYZUNIT}

Directivele şi simbolurile compilării condiţionate

Compilarea condiţionată se bazează pe evaluarea simbolurilor condiţionale.

{$DEFINE *nume*}

Introducerea simbolului "***nume***" în lista simbolurilor convenţionale. Acest simbol este recunoscut pînă nu va fi şters cu o directivă :

{$UNDEF *nume*}

Existenţa simbolului poate fi testată cu directiva **:**

{$IFDEF *nume*}

Introducerea simbolurilor condiţionale poate fi realizată şi cu opţiunea ***Conditional Defines*** (comanda ***Compiler*** din meniul ***Options***)**.**

{$UNDEF *nume*}

Ştergerea simbolului "***nume***" din lista simbolurilor condiţionale. Dacă simbolul nu este definit, directiva nu are efect.

{$IFDEF *nume*}

Dacă simbolul "***nume***" există în lista simbolurilor condiţionale, se compilează textul cuprins între această directivă şi directiva **:**

{$ELSE}
sau
{$ENDIF}
Dacă simbolul "***nume***" nu există, nu se compilează textul cuprins între această directivă şi
{$ELSE}
sau
{$ENDIF}

{$IFNDEF *nume*}
Lucrează similar directivei
{$IFDEF *nume*}
cu deosebirea că se testează lipsa simbolului "nume" în lista simbolurilor condiţionale.
{$IFOPT k+}
Dacă opţiunea **k** (care poate fi una din literele A, B, D, E, F, G, I, L, O, P, Q, R, S, T, V, X, Y) este cuplată, se compilează textul între această directivă şi directiva **:**
{$ELSE}
sau
{$ENDIF}
Dacă opţiunea k nu este cuplată, nu se compilează textul cuprins între această directivă şi
{$ELSE}
sau
{$ENDIF}

Declaraţii de variabile

Înainte de a trece la declararea variabilelor, vom arăta mai întîi modul de aşezare a programului şi a diverselor date în memoria operativă.

Programul, împreună cu procedurile şi funcţiile care aparţin programului, este depus în **segmentul de cod**. Înaintea versiunii 4.0 a compilatorului, lungimea maximă admisă a fost de **64Ko.** Prin utilizarea unit-urilor şi a bibliotecilor dinamice, lungimea maximă a devenit de **640Ko.**

În afară de **segmentul de cod**, mai există **segmentele de date**, **de stivă** şi **segmentul "extra"**. **Segmentul de date** permite memorarea datelor programului. Această memorare este ajutată şi de **segmentul de stivă**, care este zona principală a gestionării datelor dinamice. Datele programului încărcate, în funcţie de tipul lor, vor fi depuse în **segmentele de cod, stivă şi date**.

Variabilele programului principal, dacă nu sînt specificate ulterior alte atribute, sînt memorate în **segmentul de date**. Aceste variabile sînt *statice*, deoarece locaţia lor din memorie în timpul execuţiei rămîne nemodificată. Din alt

punct de vedere aceste variabile sînt şi *globale*, deoarece pot fi folosite în întreaga întindere a programului.

O altă clasă de variabile este legată de subprograme. Aceste variabile *"trăiesc"* doar în timpul execuţiei subprogramului (procedură sau funcţie). Variabilele *locale* sînt alocate la începutul segmentului de stivă. În segmentul de stivă, alocarea se face în ordine inversă, adică de la adrese mai mari la adrese mai mici. În momentul apelării subprogramului, variabilele locale vor fi alocate, adică stiva va creşte. În momentul ieşirii din subprogram, spaţiul din stivă alocat variabilelor locale, va fi eliberat. În momentul declarării variabilelor noi, spaţiul din memorie eliberat, va fi din nou reutilizabil.

Tot în *segmentul de stivă*, în zona numită ***heap***, sînt alocate ***variabilele dinamice***. Declararea ***variabilelor dinamice*** diferă substanţial de declararea celorlalte variabile, deoarece declararea lor se realizează în timpul execuţiei programului. După ce aceste variabile au fost utilizate, spaţiul ocupat de ele poate fi eliberat. Cu alte cuvinte ele ocupă spaţiu atîta timp, cît sînt efectiv necesare. Alocarea de memorie pentru variabilele dinamice se realizează cu procedurile **New** şi **GetMem**, iar eliberarea de memorie se realizează cu subprogramele **Dispose**, **FreeMem**, **Mark** şi **Release**.

Geografia memoriei este următoarea :

Adresă superioară memorie DOS		
FreePtr	Listă pentru descrierea zonelor libere în **Heap**	
HeapPtr	Zonă de memorie liberă	
HeapOrg	Heap (creşte spre adrese mari)	**OvrHeapEnd**
	Tamponul overlay (de acoperire)	**OvrHeapOrg**
Sseg:Sptr	Stivă (Stack – creşte spre adrese mici)	
Sseg:0000	Stivă neutilizată	
	Variabilele globale din programul principal şi din unit-uri	
Dseg:0000	Variabilele iniţializate (constante tipizate)	
	Segmentul de cod pentru unit-ul **SYSTEM**	
	Segmentul de cod pentru unit-ul ***U 1***	
	Segmentul de cod pentru unit-ul ***U 2***	
	Segmentul de cod pentru unit-ul ***U n***	
Cseg:0000	Segmentul de cod pentru programul principal (în care apare directiva : **Uses U 1, U 2, …, U n;**)	
PrefixSeg	Program Segment Prefix – **PSP 256 octeţi**	
Adresă inferioară memorie DOS		

Zona desemnată cu semnul 

Reprezintă conţinutul fişierului **.EXE**.

O declaraţie de variabilă asociază un nume şi un tip unei locaţii de memorie. Valorile tipului specificat vor fi memorate în locaţia respectivă.

Fiecare variabilă care apare într-un program trebuie să fie introdusă printr-o declaraţie de variabilă. Declaraţia trebuie să preceadă orice utilizare a variabilei.

Declaraţia variabilelor se face în secţiunea **var** astfel **:**

var
listă de identificatori : tip;
{…}
listă de identificatori : tip;

unde identificatorii din listă sînt despărţiţi prin virgulă. Tipul asociat unei variabile poate să fie predefinit sau utilizator,caz în care tipul respectiv trebuie să fie definit în secţiunea **type** din cadrul blocului actual sau într-un bloc exterior.

Modul Real şi modul Protejat

Microprocesoarele **80286**, **80386**, **80486** şi următoarele, pot opera în două moduri diferite **:** în **mod Real** (**R**eal **m**ode) şi în **mod Protejat** (**DPMI; DOS P**rotected **M**ode **I**nterface).

Modul Real permite gestiunea spaţiului de memorie tradiţional, cu o dimensiune de pînă la **1Mo**.

Modul Protejat se poate adresa un spaţiu de memorie de dimensiune maximă de 16 **Mo.**

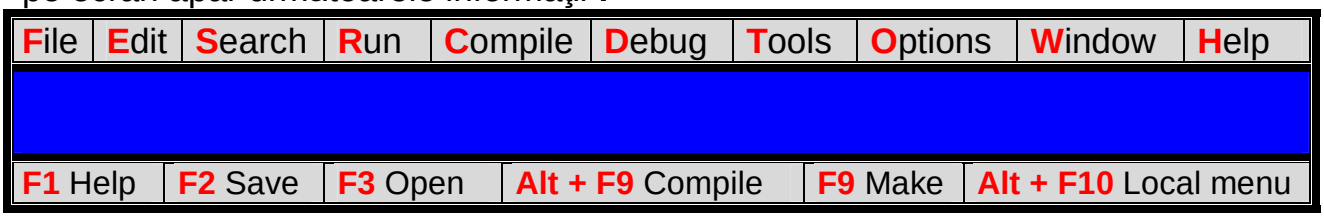
Atît la **modul real** cît şi la **modul protejat**, **adresa logică** este memorată pe două cuvinte, fiecare avînd 16 biţi, adică în total 32 de biţi. Unul dintre cuvinte este denumit **adresa de bază** (în mod real) sau **selector** (în mod protejat). Celălalt cuvînt poartă numele de **deplasament**. Din adresa logică se formează **adresa fizică**, însă prin două metode diferite. Se menţionează că **deplasamentul** în ambele cazuri, permite adresarea unei zone de memorie (segment) de 64 Ko, deoarece pe 16 biţi se pot reprezenta

$$2^{16} = 65536 = 64 * 1024 = 64 \text{ K octeţi}$$

Pornirea şi descrierea mediului de programare

Pentru a lansa compilatorul de la prompter-ul sistemului de operare, se introduce cuvîntul **Turbo** pentru **modul real** sau **BP** pentru **modul protejat**. Deoarece în modul protejat, putem accesa o memorie de pînă la **16 Mo**, în tot ceea ce vom face de acum în colo, ne vom referi la **modul protejat**.

Programul **BP.EXE** se găseşte în catalogul **C: \ BP \ BIN.** După pornire, pe ecran apar următoarele informaţii **:**



În rîndul cel mai de sus, adică în lista meniurilor, sînt specificate meniurile din mediu. Pentru a activa această linie, este suficient să se apese tasta **F10** sau se ţine apăsată tasta **Alt** plus prima literă a meniului, care este de culoare roşie sau cu mouse-ul.

Meniul File

Meniul **FILE** permite realizarea unor operaţii elementare la nivel de fişier, navigarea pe discuri, ieşirea definitivă sau parţială din mediul de programare.

New iniţiază deschiderea unei zone de lucru. Această zonă reprezentând o nouă fereastră de editare nu are în corespondenţă un fişier. Pentru recunoaşterea ferestrei, aceasta primeşte o identificare specială formată din cuvântul **NONAME** concatenat de o valoare numerică de exact două cifre între 00 şi 99. Identificarea este aleasă astfel încât să fie unică pentru ferestrele deschise şi fişierele din directorul în care se lucrează curent. Valoarea numerică este mai mare decât toate cele utilizate pentru identificările de această formă. După deschidere, fereastra devine activă, iar ferestra activă înainte de iniţierea comenzii, dacă există, devine inactivă.

Open permite încărcarea unui fişier salvat anterior cu extensia **.PAS** . Încărcarea unui fişier se poate face mai rapid prin apăsarea tastei **F3**.

Save se aplică la fereastra de editare activă şi permite realizarea salvării conţinutului pe o unitate de disc. Atunci cînd există deja un fişier salvat anterior, la o nouă salvare vechiul fişier îşi modifică extensia în **.BAK**. Salvarea se face mai rapid dacă se apasă tasta **F2**.

Save as salvează fişierul din fereastra activă sub un nume specificat, într-o unitate şi catalog specificat.

Save all salvează toate fişierele prezente în ferestrele de editare deschise şi care au fost deja modificate.

Change dir se foloseşte pentru a schimba amplasarea pe disc a zonei de lucru, operaţie cunoscută ca schimbare de director.

Print se utilizează pentru realizarea imprimării conţinutului ferestrei de editare active. Activitatea se poate activa doar de la nivelul meniului.

Printer setup se utilizează pentru stabilirea unor caracteristici legate de imprimanta conectată la calculator şi a modului de realizare a imprimării.

Dos shell se utilizează pentru ieşirea temporară din mediul de programare, cu trecerea la sistemul de operare. Pentru revenirea în mediu, trebuie dată comanda EXIT la prompterul sistemului de operare. Ieşirea temporară se face cu salvarea stării mediului la momentul ieşirii, stare care este restaurată la revenirea în mediu. Activitatea se iniţiază doar de la nivelul meniului.

Exit (Alt + X) permite ieşirea definitivă din mediul de programare şi reîntoarcerea la sistemul de operare DOS.

Meniul Edit

Comenzile meniului **E**dit permit efectuarea diferitelor operaţii legate de fişierul **Clipboard** şi anumite operaţii speciale de anulare şi de refacere a diferitelor acţiuni. Fişierul **Clipboard** realizează legătura între diferitele ferestre de editare, care la rîndul lui poate fi editat, deplasat, redimensionat sau vizualizat, la fel ca celelalte ferestre de editare.

Undo (Alt + BkSp) anulează efectul unei acţiuni anterioare sau al unui grup de acţiuni anterioare. Comanda se referă la un grup de acţiuni, dacă comutatorul **Group Undo** din meniul **Options / Environment / Editor Options** este cuplat. În caz contrar, comanda se referă doar la ultima acţiune.

Redo permite refacerea acţiunilor anulate anterior prin comanda **Undo**, în forma şi ordinea originală.

Cut (Shift + Del) decupează un bloc marcat, din fereastra activă de editare care este adăugat în clipboard. Blocul marcat, dispare.

Copy (Ctrl + Ins) copiază un bloc marcat, din fereastra activă de editare care este adăugat în clipboard. Blocul marcat rămîne intact.

Paste (Shift + Ins) inserează blocul marcat din clipboard, în fereastra activă de editare, începînd cu poziţia în care se găseşte cursorul.

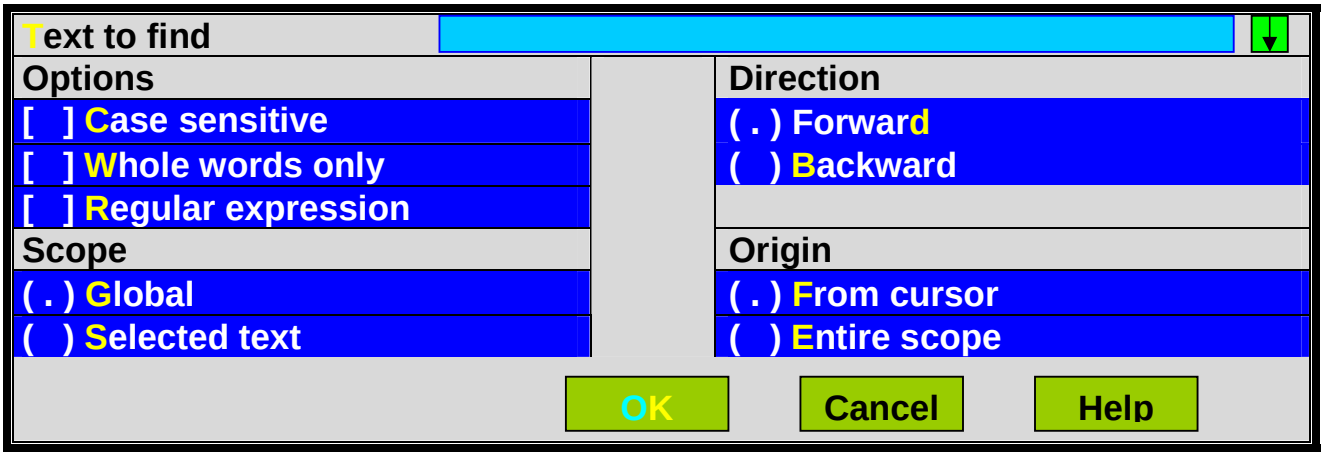
Clear (Ctrl + Del) şterge un bloc marcat din fereastra activă de editare, fără ca acest bloc să fie depus în clipboard.

Show clipboard deschide o nouă fereastră de editare cu numele **“clipboard”** şi arată conţinutul fişierului clipboard.

Meniul Search

Meniul conţine o serie de operaţii necesare deplasării într-o fereastră de editare. Deplasarea se poate realiza necondiţionat sau pentru regăsirea unei informaţii asupra căreia se poate realiza şi o editare. Meniul permite diferite căutări ale diferitelor texte, precum şi înlocuirea lor cu alte texte.

Find permite căutarea unui text şi deplasarea cursorului la textul respectiv. Se afişează o fereastră cu următoarea formă **:**



În cîmpul cu inscripţia **"Text to find"** se introduce şirul de caractere care trebuie căutat. Cîmpul este prevăzut cu o listă, cu istoricul cuvintelor căutate. Pentru demararea operaţiei de căutare se apasă butonul **"OK".** Textul găsit este marcat şi rămîne marcat pînă la următoarea acţionare a unei taste. Cursorul este plasat în urma cuvîntului găsit. În cazul în care cuvîntul nu este localizabil, se emite un mesaj de eroare de forma **"String not found"** (şirul nu este găsit).

[ ] Case sensitive – dacă este poziţionat **[X]**, se face distincţie între majuscule şi minuscule. Implicit, nu se face această distincţie.

[ ] Whole words only – dacă este poziţionat **[X]**, atunci se caută numai cuvintele "întregi", adică şirul căutat trebuie să fie încadrat de caracterul "spaţiu" sau de semne de punctuaţie.

[ ] Regular expression – dacă este poziţionat **[X]**, atunci în specificarea textului care se caută, pot fi utilizate diferite caractere funcţionale cu efect special. Aceste caractere sînt următoarele **:**

**^ $. * + [ ] **

Un caracter **^** plasat la începutul unui şir de caractere înseamnă că şirul este căutat la începutul liniei.

Un caracter **$** plasat la sfîrşitul unui şir de caractere înseamnă că şirul este căutat la sfîrşitul liniei.

Un caracter **.** semnifică faptul că în poziţia respectivă este acceptat orice caracter. De exemplu **a.b** selectează **axb, ayb,** etc.

Un caracter ***** plasat după un caracter înseamnă că acel caracter poate să se repete de un număr arbitrar de ori sau să nu apară niciodată. De exemplu, **ab*** va selecta **abb, ac,** etc.

Un caracter **+** plasat după un caracter, înseamnă că acel caracter poate să se repete de un număr arbitrar de ori, dar trebuie să apară cel puţin odată. De exemplu **ab+** va selecta **abc, abbb**, dar nu şi **ac.**

Caracterele plasate între parantezele pătrate **[** şi **]** înseamnă acceptarea oricărui caracter specificat în poziţia respectivă, dar nu şi alt caracter. De exemplu, **a[bcd]e** va selecta **abe, ace, ade.**

Caracterul **^** plasat în faţa unui şir de caractere încadrat de caracterele **[** şi **]** înseamnă negaţie **:** în poziţia respectivă este acceptat orice caracter nespecificat în listă. De exemplu, [^abc] selectează orice caracter, exceptînd caracterele **a, b, c**.

Caracterul **–** din interiorul parantezelor pătrate **[** şi **]** înseamnă limitarea domeniului. De exemplu, **[c–h]** selectează orice caracter de la **c** pînă la **g**.

Caracterul **** plasat în faţa unui caracter funcţional comunică faptul că acel caracter este tratat ca şi un caracter obişnuit. De exemplu, **\^** se referă la caracterul **^** şi nu la selectarea de la începutul liniei.

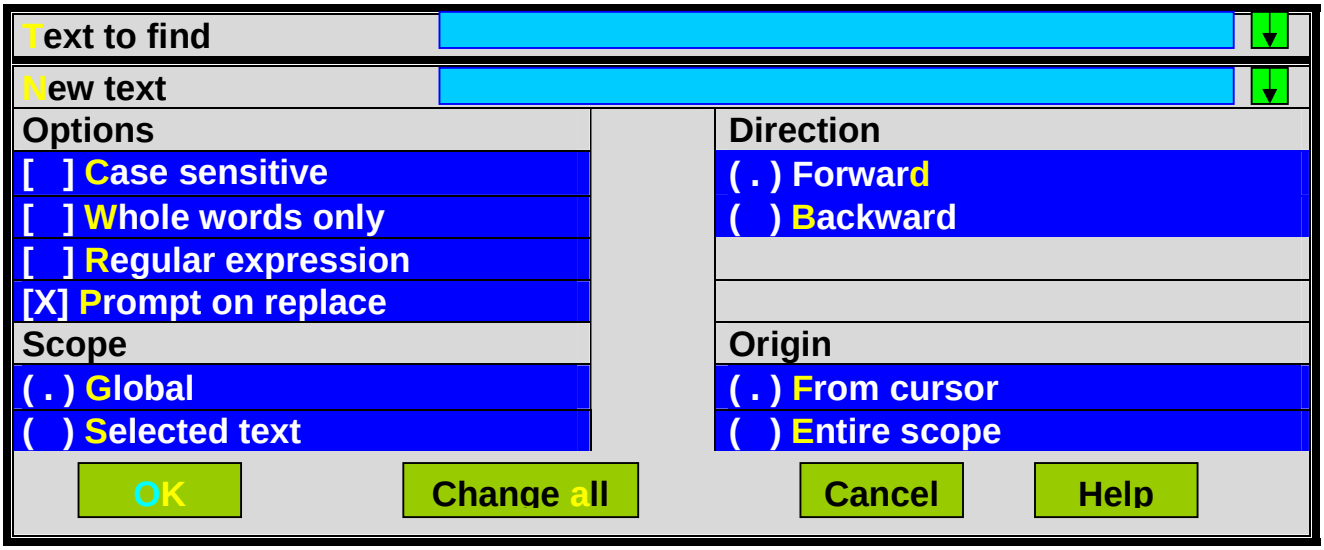
Direcţia căutării este specificată de butoanele radio din grupa cu inscripţia **"Direction"**. Dacă este poziţionat pe **(.) Forward** căutarea se face

către sfîrşitul fişierului, iar dacă este poziţionat pe **(.) Backward** căutarea se face către începutul fişierului.

Domeniul în care se efectuează căutarea este specificat de butoanele radio din grupa cu inscripţia **"Scope"**. Dacă este poziţionat pe **(.) Global**, căutarea se face în întregul fişier, în direcţia definită de **Direction**. Dacă este poziţionat pe **(.) Selected text**, căutarea este efectuată doar în blocul marcat.

Originea căutării adică punctul de unde începe căutare, este specificată cu butoanele radio, din grupa **"Origin**". Dacă este poziţionat **(.) From cursor**, căutarea se face de la poziţia actuală a cursorului, în direcţia definită de **"Direction"** şi în domeniul definit de **"Scope"**. Dacă este poziţionat **(.) Entire scope**, căutarea se face în întregul fişier sau în blocul marcat, indiferent de poziţia actuală a cursorului.

Replace permite căutare şi înlocuirea diferitelor texte. Se afişează o fereastră cu următoarea formă **:**

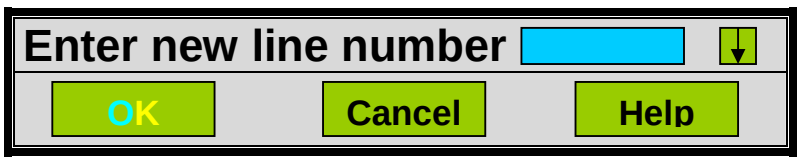


În cîmpul cu inscripţia **"Text to find"** se introduce textul care se caută iar în cîmpul cu inscripţia **"New text"** se introduce textul cu care acesta se cere a fi înlocuit.

La grupa de opţiuni, apare un nou comutator, cu numele de **"Prompt on replace"**. Dacă comutatorul este poziţionat cu **[X]**, înainte de a efectua înlocuirea textului căutat cu noul text, se va cere confirmarea efectuării înlocuirii respective de către utilizator (**Y = Da, N = Nu**). Este prezent un buton nou **"Change all"** care prin apăsare, efectuează înlocuirea automată a tuturor aparţiilor textului vechi cu textul nou.

Search again repetă ultima comandă **"Find"** sau **"Replace",** cu păstrarea tuturor opţiunilor specificate în ferestrele corespunzătoare.

Go to line number afişează o fereastră de dialog de forma **:**



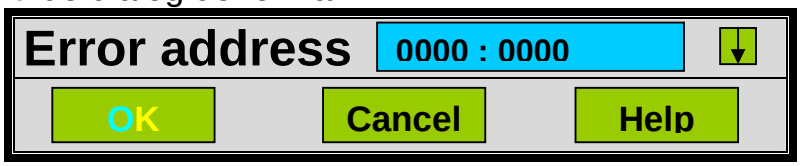
Această fereastră are ca efect deplasarea cursorului într-o linie specificată a programului. În cîmpul cu inscripţia **"Enter new line number"** se specifică numărul liniei sursă căutate. Cîmpul este prevăzut cu o comandă pentru a afişa istoricul liniilor căutate. Se aminteşte că în colţul din stînga jos al ferestrei de editare, apare numărul de linie şi de coloană pentru poziţia actuală a cursorului.

Show last compiler error afişează din nou codul şi textul mesajului de eroare, corespunzător ultimei erori de compilare. Cursorul va fi deplasat în dreptul locaţiei erorii. Dacă compilarea s-a realizat fără erori sau dacă programul sursă nu a fost compilat, comanda nu poate lucra.

Find error permite localizarea unei erori de execuţie. Cînd se execută sub sistemul **DOS** un fişier executabil, o eroare de execuţie apare sub forma **:**

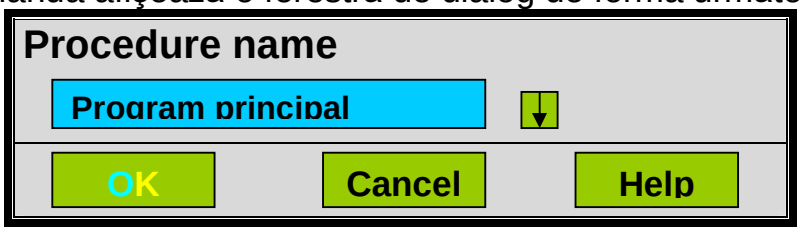
Runtime error Cod_de_eroare **at** seg **:** depl

unde **seg** şi **depl** reprezintă adresa segmentului şi a deplasamentului instrucţiunii care a declanşat eroarea. După ce se notează aceste valori, se încarcă programul sursă într-o fereastră de editare sau se specifică numele programului primar, în comanda **"Primary file"** din meniul **"Compile"**. În continuare se fixează destinaţia compilării pe disc şi se apelează comanda **"Find error"**. Comanda afişează o fereastră de dialog de forma **:**



În cîmpul de introducere cu inscripţia **"Error address"** se înscrie adresa şi deplasamentul erorii respective, de exemplu **00A5 : 000F.** După ce comanda a fost executată, cursorul va fi deplasat la începutul instrucţiunii sursă care a cauzat eroare respectivă. Linia **"Error address"** este prevăzută cu comanda pentru afişarea listei cu istoricul căutărilor. Comanda **"Find error"** este activabilă numai atunci cînd este poziţionat cu **[X]** comutatorul **"Integrated Debugging"** din meniul **Options**, submeniul **Debugger**.

Find procedure poate fi utilizată numai atunci cînd programul sursă a fost deja compilat. Ea serveşte la căutarea locaţiei unde este definită o procedură sau funcţie. Comanda afişează o fereastră de dialog de forma următoare **:**



În cîmpul cu inscripţia **"Procedure name"** se specifică numele subprogramului căutat. Numele astfel căutat va fi căutat în întregul fişier sursă, inclusiv unit-urile şi fişierele sursă. Cursorul va fi poziţionat pe numele subprogramului căutat. Deci comanda nu se referă la execuţia codului, ci doar la vizualizarea locaţiei de definiţie a subprogramului. Comanda **Find procedure** diferă de comanda **Find** a meniului **Search** prin faptul că prima localizează definiţia subprogramului iar cea dea doua, locul în care este apelat subprogramul respectiv.

Versiunea în mod protejat a compilatorului, adică programul **BP.EXE**, permite afişarea ferestrelor de tip **"browse"**, adică un studiu amănunţit despre toate simbolurile utilizate în fişierul sursă. Aceste simboluri pot fi obiecte, unit-uri, variabile, tipuri, constante, etichete globale şi locale, subprograme de tip procedură şi funcţie. Comenzile de tip **"browse"** pot fi aplicate numai programelor deja compilate. Informaţiile afişate în aceste ferestre conţin fie informaţii de tip tabelă de materii, pentru obiecte, unităţi şî simboluri globale ale programului sursă, fie informaţii despre un simbol selectat.

Meniul Run

Comenzile meniului **Run** permit execuţia unui program, reiniţializarea execuţiei, execuţia programului pînă la poziţia actuală a cursorului, execuţia pas cu pas precum şi stabilirea parametrilor din linia de comandă.

Run (Ctrl + F9) execută un program. Dacă este nevoie, sînt folosiţi parametrii din linia de comandă, specificaţi în comanda **"Parameters"** din meniul **"Run".** Dacă programul sursă a fost modificat de la ultima compilare sau dacă programul nu a fost în prealabil compilat, se trece automat prima dată la compilare, apoi la execuţie. Programul este executat pînă la sfîrşitul programului sau pînă la un punct de întrerupere, dacă există un astfel de punct. La terminarea execuţiei programului, controlul este redat mediului de programare.

Step over (F8) lucrează similar comenzii **"Trace into"**, cu următoarea deosebire **:** dacă instrucţiunea actuală este un apel de subprogram, întregul subprogram este executat într-un singur pas şi bara de execuţie va fi poziţionată pe instrucţiunea care urmează după apelul de subprogram.

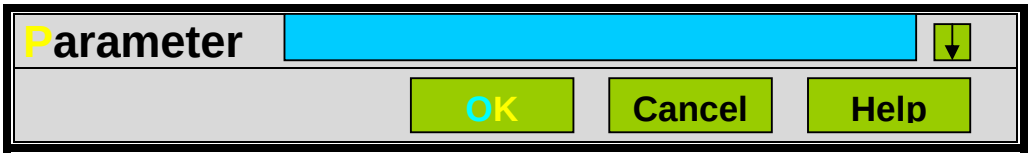
Trace into (F7) execută instrucţiunea următoare din program, iar la începutul execuţiei va fi executată prima instrucţiune executabilă a programului. Dacă instrucţiunea următoare este un apel la un subprogram, atunci execuţia va continua cu prima instrucţiune executabilă a subprogramului. Aplicînd repetat această comandă, se poate efectua execuţia pas cu pas, a programului. Se menţionează că instrucţiunea care urmează să fie executată, este evidenţiată cu o bară, numită **"bară de execuţie".** Comanda poate fi utilizată numai atunci cînd în momentul compilării a fost cuplată cheia **"Debug information"** din meniul **"Options"**, comanda **"Compiler"**, grupa **"Debugging"**, respectiv directiva **{$D+}.** În caz contrar, se va emite un semnal de eroare, de forma **"No debug info for program entry point. Run anyway?"**, adică nu există informaţii de depanare.

Go to cursor (F4) începe sau continuă execuţia unui program din poziţia curentă de execuţie, pînă la linia în care se găseşte cursorul. Dacă nu s-a început încă depanarea programului, poziţia curentă de execuţie este linia în care se găseşte prima instrucţiune executabilă. Comanda poate fi utilizată numai atunci cînd în momentul compilării a fost cuplată cheia **"Debug information"** la fel ca la comanda de mai sus. Altfel, se emite un mesaj de eroare, de forma **"No code generated for this line"**. Această comandă , nu creează un punct permanent de întrerupere.

Program reset (Ctrl + F2) reiniţializează actuala sesiune de depanare. Se eliberează memoria alocată programului şi se închide fiecare fişier

deschis, dar valorile variabilelor rămîn nemodificate., adică nu sînt reiniţializate. Punctele de întrerupere nu sînt anulate, dar la următoarea execuţie, programul va fi reluat de la început.

Parameters afişează o fereastră de dialog de forma **:**



În cîmpul cu inscripţia **"Parameter"** se pot introduce parametrii liniei de comandă a programului. Pentru introducere se poate folosi şi lista cu istoricul parametrilor introduşi anterior. În interiorul programelor, numărul lor poate fi determinat cu funcţia **"ParamCount"** iar valoarea celui de –al **"N-lea"** parametru, poate fi determinată cu funcţia **"ParamStr(n)".** Pentru valoarea **N = 0** funcţia va returna numele programului. Dacă nu există nici un parametru, atunci funcţia **"ParamCount"** va returna valoarea 0.

Meniul Compile

Comenzile meniului **Compile** permit compilarea programelor şi a unit-urilor.

Compile (Alt + F9) permite compilarea fişierului care se găseşte în fereastra actuală de editare. Acest fişier poate fi atît un program principal, cît şi un unit sau bibliotecă de legare dinamică (**.DLL**). Textul compilat poate să conţină şi referiri la fişiere externe de includere (directive de tip **$I nume**).

Make (F9) compilează fişierul primar specificat, iar în lipsa unui astfel de fişier, se compilează fişierul din fereastra actuală de editare. Comanda determină unit-urile folosite de programul primar. Acele unit-uri care sînt prezente numai în format sursă, vor fi compilate automat şi vor fi generate fişiere de extensie **.TPU** corespunzătoare. De exemplu, dacă pentru un unit **U1** se găseşte fişierul **U1.Pas**, acest unit este compilat şî rezultatul este depus în fişierul **U1.TPU** Dacă pentru un unit sînt prezente atît fişierul cu extensia **.PAS** cît şi fişierul cu extensia **.TPU**, dar data de creare a fişierului cu extensia **.PAS** este mai recentă decît data de creare a fişierului cu extensia **.TPU**, adică dacă după generarea fişierului **.TPU** a mai fost actualizat fişierul sursă **.PAS**, unit-ul respectiv va fi recompilat şi astfel va fi regenerat şi fişierul **.TPU.**

Build compilează fişierul primar şi toate unit-urile utilizate de acesta. Recompilarea unit-urilor nu este condiţionată de data de creare a acestora. Aceată comandă este similară cu comanda **Make** cu deosebirea că compilarea este independentă de data de creare. La comanda **Make**, recompilările se referă doar la acele unit-uri, care nu mai sînt actuale.

Target permite selectarea platformei de destinaţie, adică aplicaţie în mod **Real**, în mod **Protejat** sau de tip **Windows.**

Primary file permite specificarea fişierului primar, adică a fişierului care va fi compilat cu comanda **Make** sau **Build**, indiferent de conţinutul ferestrei actuale de editare.

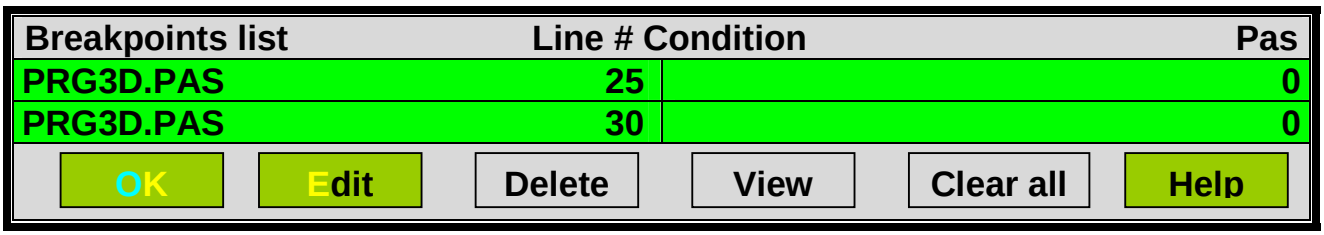
Clear primary file şterge fişierul primar care a fost actual în momentul actual.

Information afişează informaţii despre fişierul compilat, adică numărul liniilor compilate, dimensiunea codului generat, dimensiunea segmentului de date şi de stivă, a heap-ului minim şi maxim şi despre memorie.

Meniul Debug

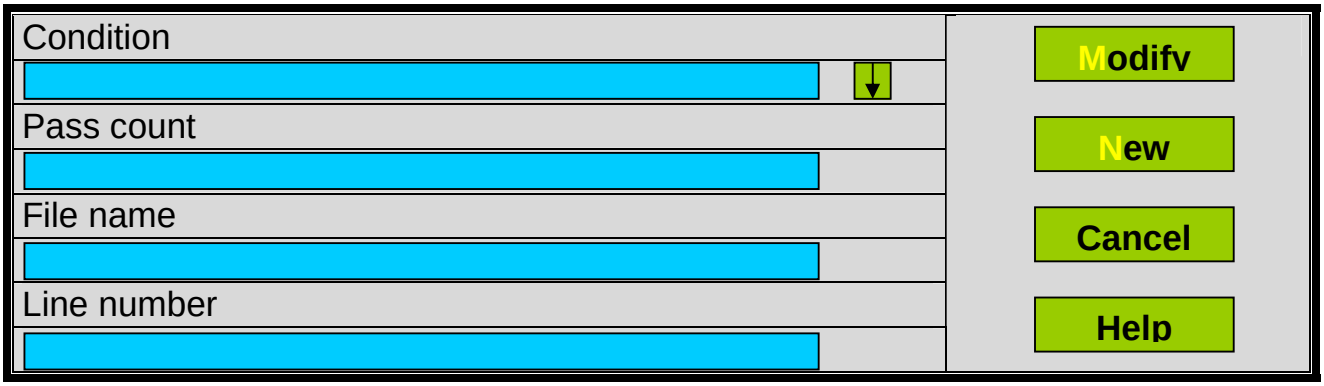
Comenzile meniului **Debug** (depanare) permit crearea şi ştergerea unor puncte de întrerupere, evaluarea şi modificarea valorilor unor expresii numite expresii de **"urmărire"**.

Breakpoints afişează o fereastră care permite vizualizarea punctelor de întrerupere instalate precum şi modul lor de utilizare.



Această fereastră conţine o listă cu următoarele informaţii **:** pentru fiecare punct de întrerupere apare numele programului sursă, numărul liniei sursă, condiţia care trebuie să fie îndeplinită pentru ca să aibe loc oprirea la punctul respectiv precum şi un contor de trecere. Valorile posibile sînt în intervalul **: {0 … 2^{15} – 1 = 32767 }** Acest contor precizează numărul trecerilor care încă nu declanşează oprirea programului la instrucţiunea respectivă. La fiecare trecere este testat un contor interior care a fost iniţializat cu valoarea contorului de trecere. Dacă valoarea este mai mare decît zero, contorul este decrementat cu unu şi programul funcţionează mai departe. Dacă valoarea contorului, este zero, programul se opreşte. În coloana care precizează condiţia, se introduce o condiţie, iar cînd aceasta este adevărată, programul se opreşte. Fereastra de dialog **"BreakPoints"** este prevăzută cu cu butoanele standard **"Ok"** şi **"Help"**, dar şi cu butoanele de comenzi **"Edit, Delete, View** şi **Clear All".**

Edit deschide o nouă fereastră de dialog cu titlul **"Edit Breakpoint"** ca mai jos.



Această fereastră, permite editarea punctelor de întrerupere existente. În această fereastră se pot modifica **:**

- condiţia care cauzează declanşarea întreruperii, cu listă de istoric
- contorul de trecere, ataşat întreruperii
- numele fişierului la care se referă întreruperea
- numărul liniei sursă a întreruperii

Modificările efectuate sînt validate prin butonul de comandă **Modify**. Pentru definirea unui nou punct de întrerupere, care poate fi în cadrul aceluiaşi program se apasă butonul **New**. Celelalte comenzi ale ferestrei **"Breakpoints"**, sînt **:**

Delete care şterge punctul actual de întrerupere, din lista punctelor de întrerupere.

View care caută în textul sursă, punctul actual de întrerupere şi poziţionează cursorul, în dreptul liniei respective.

Clear all care şterge toate punctele de întreruperi din listă.

Call stack (Ctrl + F3) afişează o fereastră care vizualizează o stivă cu istoricul apelurilor subprogramelor. Sînt afişate numele subprogramelor şi parametrii actuali corespunzători prin utilizarea cărora s-a ajuns la poziţia actuală din program. Închiderea ferestrei se poate face cu mouse-ul sau cu comanda **Alt + F3.**

Register afişează o fereastră în colţul din dreapta sus al ecranului, ce nu poate fi redimensionată dar poate fi mutată, cu numele **CPU** şi care prezintă conţinutul registrilor microprocesorului. Fereastra se foloseşte la depanarea secvenţelor de program scrise în limbajul de asamblare.

Watch afişează fereastra cu titlul **"Watches"**, în interiorul căreia sînt afişate valorile actuale ale variabilelor şi expresiilor supravegheate. Pentru a vedea valoarea unei variabile, după ce programul este oprit prin terminare sau punct de întrerupere, se apasă tasta **"Insert"** şi apoi se scrie numele variabilei şi se apasă **"Enter"**.

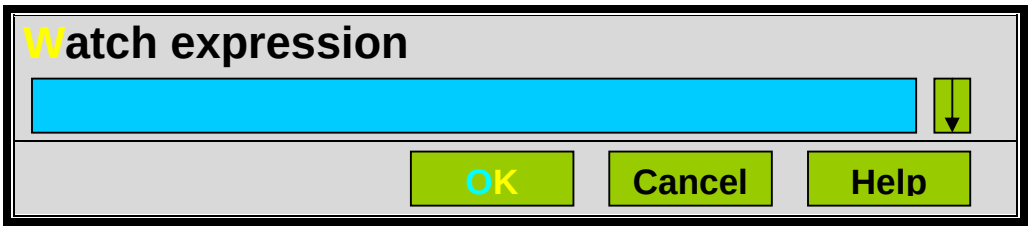
Output activează o fereastră care afişează textul din linia de comandă **DOS** şi textul generat de programul care se execută. Pot fi vizualizate doar informaţiile în mod text, iar cele în mod grafic nu. Fereastra este utlizată pentru că pot fi urmărite simultan, atît programul sursă, cît şi rezultatele care se afişează.

User screen (Alt + F5) afişează un ecran întreg cu toate informaţiile trimise de programul actual, indiferent dacă informaţiile sînt în mod text sau în mod grafic. Fereastra este statică şi nu poate fi redimensionată sau mutată.

Evaluate/modify (Ctrl + F4) permite evaluarea şi vizualizarea valorii unor variabile şi expresii, precum şi modificarea valorii unei variabile. În cîmpul cu inscripţia **"Expression"** se poate introduce orice variabilă sau expresie, pentru care se doreşte evaluarea valorii actuale. În momentul lansării comenzii, este introdus automat simbolul de la poziţia cursorului.

După ce se introduce expresia în cîmpul **"Expression"**, se acţionează butonul de comandă **"Evaluate"**. În acest moment, valoarea expresiei este afişată în cîmpul cu inscripţia **"Result"**. Valoarea afişată poate fi modificată la o altă valoare.

Add watch (Ctrl + F7) permite definirea unei variabile sau expresii a cărei valoare este supravegheată permanent. Se afişează o fereastră ca mai jos.



În cîmpul cu inscripţia **"Watch expression"** se introduce numele variabilei sau expresia care se urmăreşte. Comanda poat fi utiizată doar atunci cînd în momentul compilării este cuplată cheia **"Debug information"** din meniul **"Options**, comanda **Compiler**, grupa **Debugging"** sau directiva **{$D+}.**

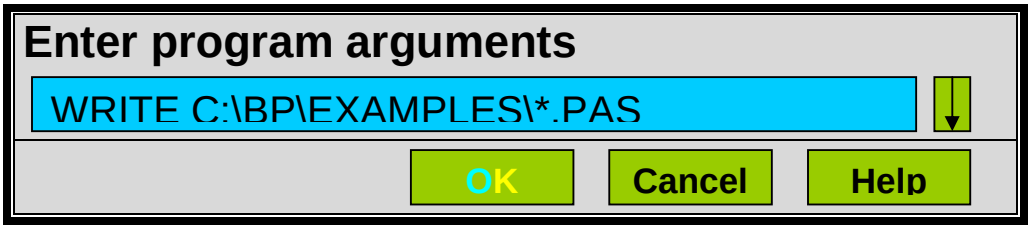
Add breakpoint permite introducerea punctelor de întrerupere noi şi specificarea condiţiilor de oprire. Se menţionează că un program se poate opri cu comanda **Ctrl + Break**, care este o ieşire obişnuită în cazul unui program care are un ciclu infinit.

Meniul Tools

Meniul **Tools** permite lansarea în execuţie a diferitelor programe externe din interiorul mediului de programare **7.0** . Numele programelor care pot fi lansate în execuţie au regim identic cu cel al comenzilor unui meniu. Pentru lansarea lor se activează comanda respectivă.

Messages afişează o fereastră cu acelaşi nume, în partea inferioară a ecranului, care permiteafişarea mesajelor acelor programe care transmit mesajele către mediul de programare prin intermediul unui filtru, de exemplu utilitarul **"Grep"** sau **"Turbo Assembler"**.

Grep este un utilitar care permite căutarea diferitelor cuvinte în fişierele text. De exemplu, dacă se doreşte căutarea tuturor liniilor sursă care conţin cuvîntul **"Write"** în toate fişierele cu extensia **".Pas"** din catalogul **"C:\BP\Examples"**, atunci se activează comanda **"Grep"**. Apare o fereastră în care se introduce cuvîntul căutat şi calea catalogului în care se caută, separate de un spaţiu.



Chiar dacă nu a fost activată anterior, apare după apăsarea butonului **"OK"** apare fereastra **"Messages"**, care afişează o listă a tuturor apariţiilor cuvîntului căutat, împreună cu numărul de linie din program în care a fost găsit.

Go to next (Alt + F8) asigură trecerea la următoarea linie a ferestrei de mesaje. Dacă această linie reperează o linie dintr-un fişier text, atunci fişierul

va fi deschis, dacă nu a fost deschis în prealabil şi cursorul este poziţionat la linia referită a fişierului.

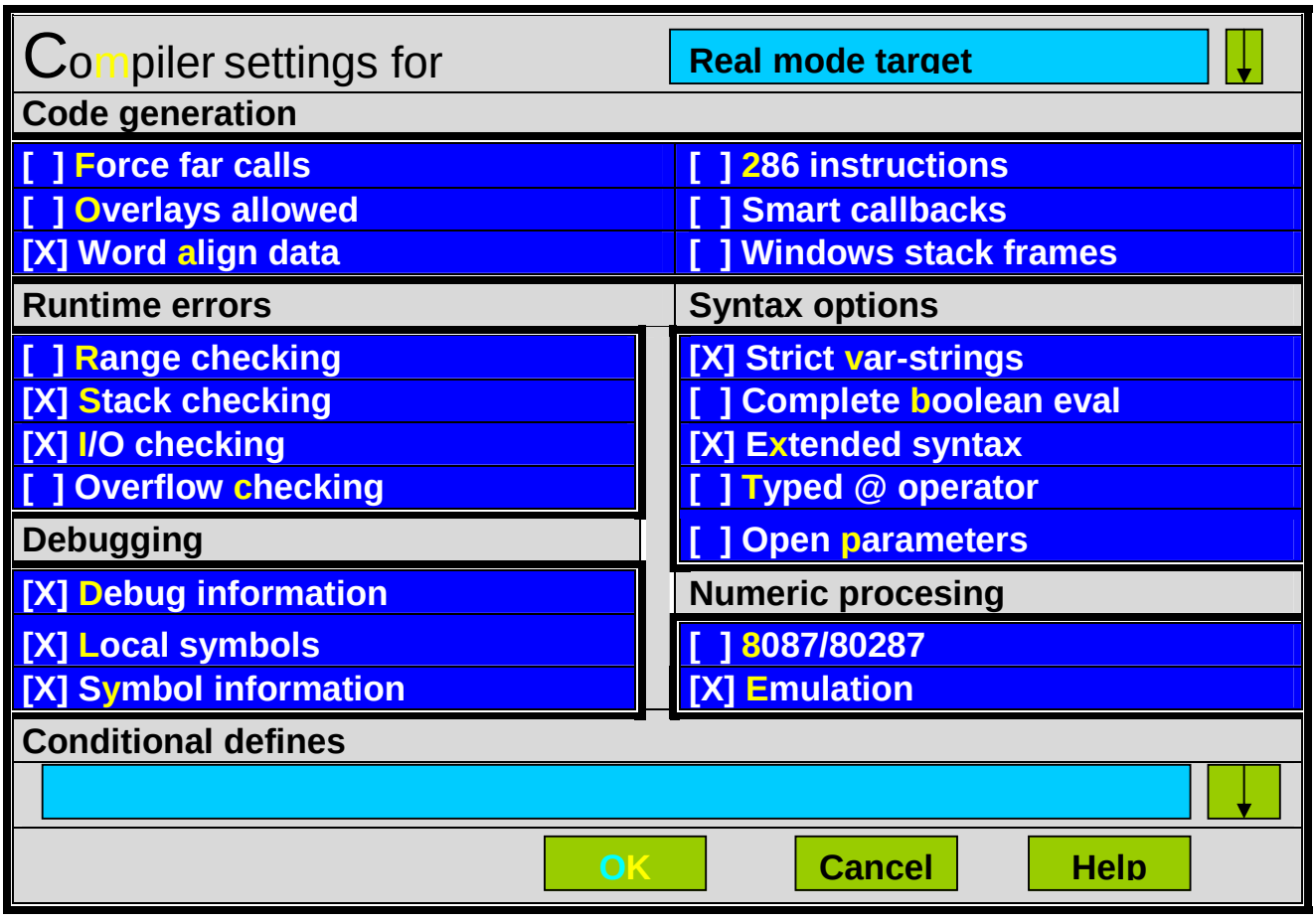
Go to previous (Alt + F7) este analoagă comenzii precedente, cu deosebirea că se trece la linia precedentă a ferestrei de mesaje.

În meniul **"Tools"** pot fi introduse şi alte utilitare sau cele existente pot fi şterse. Ele pot fi realizate cu ajutorul comenzii **"Tools"** a meniului **"Options"**.

Meniul Options

Comenzile meniului **"Options"** permit fixarea diferitelor opţiuni referitoare la compilator, editor de legături, depanator, etc.

Compiler afişează o fereastră de dialog în care opt fi precizate opţiunile care guvernează tipul generării de cod, tipul tratării erorilor de execuţie, tipul opţiunilor sintactice, tipul gestionării numerelor reale, cantitatea informaţiilor de depanare precum şi definirea simbolurilor condiţionate.



Cîmpul cu titlul **"Compiler settings"** apare numai la compilatorul **"BP.EXE"**. În acest cîmp se poate selecta natura aplicaţiei pentru care se fac precizări de opţiuni de compilare. Valorile posibile sînt **:**

- aplicaţie în mod **R**eal (Real mode target)
- aplicaţie în mod **P**rotejat (Protected mode target)
- aplicaţie sub **W**indows (Windows target)
- aplicaţii pentru toate cele trei platforme de destinaţie (All target)

Grupa "Code generation" se referă la comutatorii care definesc modul de generare a codului.

[ ] Force far calls

Dacă comutatorul nu este poziţionat, pentru salvarea adresei de retur din subprograme se foloseşte modelul de apel apropiat (**near**, 2 octeţi pentru deplasament). Cu comutatorul poziţionat se foloseşte modelul de apel îndepărtat (**far**, pe 4 octeţi, 2 octeţi pentru deplasament şi 2 octeţi pentru adresă de segment). Este echivalent cu directiva de compilare **{$F+}** sau **{$F-}**.

[ ] Overlays allowed

Dacă comutatorul nu este poziţionat, nu se va folosi modul de generare tip acoperire, adică un unit nu poate fi acoperit de un alt unit. Este echivalent cu directiva de compilare **{$O+}** sau **{$O-}**.

[ ] Word align data

Dacă comutatorul este poziţionat, se va alege modul de aliniere tip cuvînt, adică toate variabilele de lungime mai mare de un octet sînt aliniate la adrese de cuvînt (adrese pare). Dacă este necesar, între variabile sînt inseraţi octeţi neutilizaţi. Acest tip de aliniere majorează viteza de acces la variabile, cu preţul pierderii de spaţiu de memorie. Cu comutatorul poziţionat se alege modul de aliniere tip octet, adică variabilele sînt memorate la prima adresă liberă, indiferent de dimensiunea lor. Este echivalent cu directiva de compilare **{$A+}** sau **{$A-}**.

[ ] 286 instructions

Dacă comutatorul este poziţionat, codul generat conţine instrucţiunile procesorului 80286, mărind astfel viteza de execuţie. Acest cod însă nu se mai poate executa pe procesoarele 8088 sau 8086. Este echivalent cu directiva de compilare **{$G+}** sau **{$G-}**.

[ ] Smart callbacks xxx

[ ] Windows stack frames xxx

Grupa Runtime errors se referă la comutatori care guvernează tipul tratării erorilor de execuţie.

[ ] Range checking

Dacă comutatorul este poziţionat, se testează, se testează apartenenţa în domeniul permis de tipul de enumerare sau interval a diferitelor variabile, valabilitatea indicilor de tablou şi de şir de caractere. În caz de eroare se afişează un mesaj şi programul se opreşte. Este echivalent cu directiva de compilare **{$R+}** sau **{$R-}**.

[ ] Stack checking

Dacă comutatorul este poziţionat, la fiecare apel de subprogram se testează dacă în stivă este loc suficient pentru memorarea variabilelor locale. Dacă spaţiul este insuficient, programul se opreşte. Este echivalent cu directiva de compilare **{$S+}** sau **{$S-}**.

[ ] I / O checking

Dacă comutatorul este poziţionat, la fiecare operaţie de intrare / ieşire se testează dacă a apărut o eroare. În caz afirmativ, se afişează un mesaj de eroare şi programul se opreşte. Dacă comutatorul nu este poziţionat, eroarea apărută poate fi tratată în program cu funcţia **IOResult.** Este echivalent cu directiva de compilare **{$I+}** sau **{$I-}**.

[ ] Overflow checking

Dacă comutatorul este poziţionat, se va testa depăşirea rezultatelor operaţiilor **"+,–, * "** asupra valorilor întregi, precum şi a rezultatelor funcţiilor **"Abs, Sqr, Succ** şi **Pred"**. Rezultatele funcţiilor **"Inc** şi **Dec"** nu sînt verificate. Este recomandată activarea simultană cu comutatorul **"Range checking"**. Este echivalent cu directiva de compilare **{$Q+}** sau **{$Q-}**.

Grupa Syntax options se referă la opţiunile sintactice.

[ ] Strict var–strings

Dacă comutatorul este poziţionat, la transmisia parametrilor variabili de tip şir de caractere, se compară lungimea parametrului formal. Dacă lungimile sînt diferite, se semnalează o eroare de execuţie şi programul se opreşte. Este echivalent cu directiva de compilare **{$V+}** sau **{$V-}**.

[ ] Complete boolean eval

Dacă comutatorul este poziţionat, fiecare termen dintr-o expresie logică este evaluat. Dacă comutatorul nu este poziţionat, se fac optimizări în codul generat. Este echivalent cu directiva de compilare **{$B+}** sau **{$B-}**.

[ ] Extended syntax

Dacă comutatorul este poziţionat se permite ca funcţiile să fie apelate ca şi cum ar fi proceduri, iar rezultatul returnat de funcţie este neglijat. Sintaxa extinsă însă nu poate fi aplicată pentru funcţiile unit-ului **System**. Utilizarea funcţiilor ca şi proceduri are sens, de exemplu cu subprogramele unit-ului **Strings** sau în context cu funcţia **Readkey**, cînd nu prezintă interes codul tastei apăsate. Este echivalent cu directiva de compilare **{$X+}** sau **{$X-}**.

[ ] Typed @ operator

Dacă comutatorul este poziţionat, atunci se verifică tipul reperului generat de operatorul de adresare @. Dacă comutatorul nu este poziţionat, atunci tipul operatorului @ va fi un reper fără tip de bază **:** pointer. Este echivalent cu directiva de compilare **{$T+}** sau **{$T-}**.

[ ] Open parameters

Dacă comutatorul este poziţionat, atunci la un parametru formal de tip **string** poate să lipsească specificarea exactă a lungimii şirului de caractere, iar parametrul actual corespunzător poate să fie de tip **string** de lungime arbitrară (parametru tip deschis).Este echivalent cu directiva de compilare **{$P+}** sau **{$P-}**.

Grupa Debugging se referă la cantitatea informaţiilor de depanare păstrate.

[ ] Debug information

Dacă comutatorul este poziţionat, se generează o tabelă cu numărul liniilor sursă pentru fiecare instrucţiune din program. În această tabelă se face corespondenţă între adresa codului generat şi numărul liniei sursă corespunzător. Astfel mediul permite execuţia pas cu pas, pînă la locaţia cursorului precum şi definirea punctelor de întrerupere. Dacă comutatorul nu este poziţionat, necesităţile de memorie descresc, dar mijloacele de depanare devin inaccesibile. Este echivalent cu directiva de compilare **{$D+}** sau **{$D-}**.

[ ] Local symbols

Dacă comutatorul este poziţionat, se generează o tabelă cu numele şi tipul tuturor variabilelor şi constantelor locale dintr-un modul. Astefel mediul va permite evaluarea şi modificarea valorilor variabilelor locale, precum şi istoricul apelurilor de subprograme. Funcţiile **Linker / Map file** şi **Debugger / Standalone** ale meniului **Options** pot să livreze informaţii referitoare la variabilele locale numai pentru acele subprograme care au fost compilate cu comutatorul **"Local symbols"** poziţionat. Este echivalent cu directiva de compilare **{$L+}** sau **{$L-}**.

[ ] Symbol information

Comutatorul poate fi utilizat numai la compilatorul **"BP.EXE"**. Dacă comutatorul este poziţionat, compilatorul introduce în codul generat informaţii de tip **"browse"**. În acest caz se permite afişarea diferitelor ferestre de tip **"browse"** referitoare la obiecte, unit-uri, variabile globale şi simboluri. Este echivalent cu directiva de compilare **{$Y+}** sau **{$Y-}**. Cînd comutatorul **"Symbol information"** este cuplat, atît comutatorul **{$D+}** cît şi comutatorul **{$L+}** trebuie să fie cuplat.

Grupa Numeric procesing se referă la comutatori care guvernează tipul gestionării numerelor reale.

[ ] 8087 / 80287

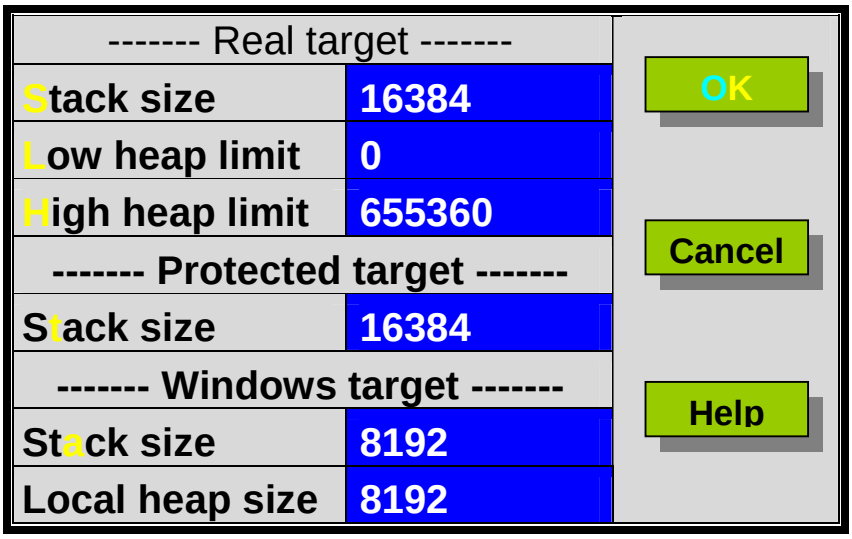
Dacă comutatorul este poziţionat, atunci se permite utilizarea tuturor tipurilor reale **(real, single, double, extended, comp)**. Dacă comutatorul nu este poziţionat, atunci se permite doar utilizarea tipului **"real"**, celelalte tipuri reale sînt interzise. Este echivalent cu directiva de compilare **{$N+}** sau **{$N-}**.

[ ] Emulation

Dacă comutatorul este poziţionat, atunci se utilizează o bibliotecă care emulează coprocesorul matematic 8087, dacă acest coprocesor nu este prezent. Fişierul executabil poate fi utilizat de orice echipament, indiferent dacă 8087 este sau nu prezent. Este echivalent cu directiva de compilare **{$E+}** sau **{$E-}**.

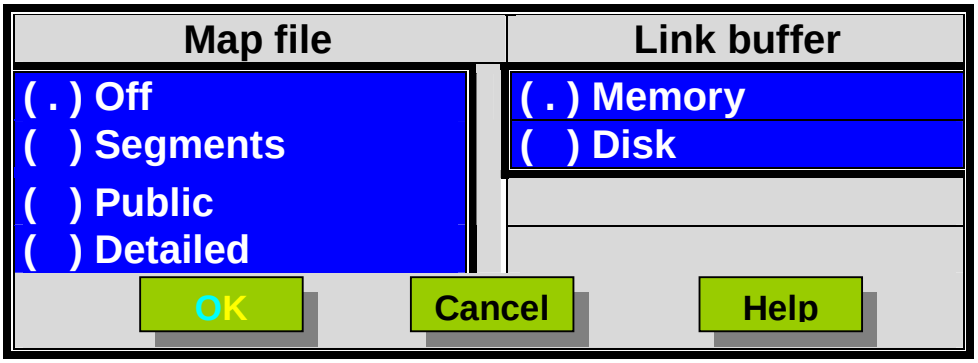
Memory sizes deschide o fereastră de dialog în care pot fi precizate necesitaţile de memorie ale unui program.

Fereastra are forma de mai jos **:**



Se observă că se poate specifica dimensiunea stivei **(Stack size)** şi specificarea dimensiunii minime **(Low heap limit)** şi maxime **(High heap limit)** pentru zona **heap**. Este echivalent cu directiva de compilare **{$M+}** sau **{$M-}**. Valorile implicite sînt **: {$M 16384,0,655360}.**

Linker afişează o fereastră care se referă la precizarea informaţiilor necesare editorului de legături integrat. Fereastra are forma **:**



Butoanele radio din grupa cu titlul **“Map file”** se referă la fişierul text de extensie **“.MAP”** care este un fişier hartă de memorie creat de editorul din mediul de programare. Acest fişier conţine informaţii necesare depanării programului şi el este depus pe disc, în catalogul specificat de cîmpul **“Object directories”** din meniul **“Options”** comanda **“Directories”**.

(.) Off – nu se creează fişierul **“ *.MAP”**

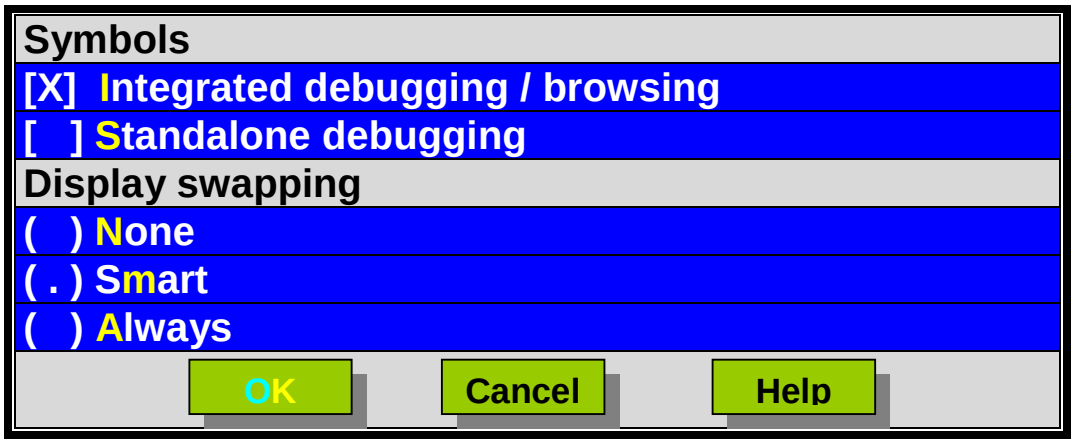
( ) Segments – fişierul **“ *.MAP”** va conţine doar informaţiile referitoare la segmente **:** nume, dimensiune, adresa de început şi de sfîrşit, clasa.

( ) Public – fişierul **“ *.MAP”** va conţine şi tabela care conţine numărul liniilor sursă şi numele modulelor, precum şi adresa în segmentul de cod a instrucţiunilor corespunzătoare liniilor. La aceste module, la care în momentul compilării, comutatorul local **“Local symbols”** este cuplat **{$L+}** (meniul **Options**, comanda **Compiler**), în fişierul **“ *.MAP”** se vor depune informaţii şi despre simbolurile locale. Butoanele radio din grupa cu titlul **“Link buffer”** stabilesc locul tamponului editorului de legături.

(.) Memory – Tamponul este rezervat în memorie

( ) Disk – Tamponul este rezervat pe disk.

Debugger deschide o fereastră de dialog în care pot fi precizate opţiunile referitoare la depanator şi modul de utilizare a ecranului.



[X] Integrated debugging / browsing

Dacă este cuplat, în codul generat vor fi incluse informaţii, care permit execuţia programului în regim de depanare cu ajutorul depanatorului integrat.

[ ] Standalone debugging

Dacă este cuplat, în codul generat vor fi incluse informaţii, care permit depanarea cu ajutorul utilitarului **“Turbo Debugger”**.

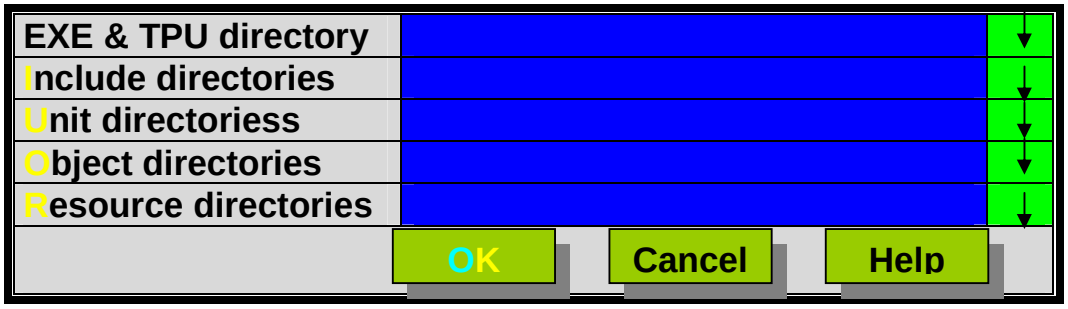
Butoanele radio din grupa cu titlul **“Display swapping”** determină modul de schimbare şi utilizare a ecranului.

( ) None – Dacă un program este executat din mediul integrat, programul va suprascrie ecranul mediului, adică nu are loc o schimbare de ecran. Ecranul mediului poate fi restabilit cu comanda **“Refresh display”** a mediului **“Window”**.

(.) Smart – În momentul scrierii pe ecran de către programul executat, ecranul mediului este schimbat pentru o scurtă perioadă necesară scrierii pe ecranul utilizatorului, după care se face revenirea la starea anterioară pornirii programului.

( ) Always – După fiecare instrucţiune are loc o schimbare de ecran.

Directories afişează o fereastră de dialog în care pot fi precizate diferite directoare care conţin anumite fişiere necesare mediului de programare.



- **EXE & TPU directory** – este catalogul care conţine fişierele **“.EXE, .TPU, .MAP”**. Dacă nu se introduce nimic în cîmpul aferent, aceste fişiere sînt depuse în catalogul fişierului sursă.
- **Include directories** – se referă la directoarele care conţin fişierele de includere ale utilizatorului. În programul sursă, aceste fişiere sînt referenţiate cu directiva **{$Inume_fişier}**. În cazul directoarelor multiple, acestea sînt separate cu **“;”**.
- **Unit directories** – se referă la directoarele care conţin fişierele unit-urilor standard sau unit-urilor utilizator.
- **Object directories** – se referă la directoarele care conţin fişierele obiect **“.OBJ”** ale utilizatorului adică codul obiect al subprogramelor scrise în limbaj de asamblare cu extensia **“.ASM”.** În programul sursă, aceste fişiere sînt referenţiate cu directiva **{$Lnume_fişier}.**

Browser poate fi utilizată numai în cazul compilatorului **“BP.EXE”**.

Symbols	
[X] Labels	[X] Variables
[X] Constants	[X] Procedures
[X] Types	[ ] Inherited
Sub-browsing	**Prefered pane**
(.) New browser	(.) Scope
( ) Replace curent	( ) Reference
Display	
[X] Qualified symbols	[ ] Sort always
OK Cancel Help	

Grupa cu titlul **“Symbols”** defineşte natura simbolurilor pentru care se doreşte aplicarea serviciilor **“browse”**, adică simbolurile care pot apare în ferestrele de tip **“browse”**. Există şase categorii de simboluri utilizabile **:** etichete **(Labels)**, constante **(Constants)**, tipuri **(Types)**, variabile **(Variables)**, proceduri **(Procedures)** şi simboluri moştenite **(Inherited)**.

Grupa cu titlul **“Sub-browsing”** conţine următoarele butoane radio **:**

(.) New browser – adică dacă din fereastra actuală de tip **“browse”** se trece la afişarea informaţiilor mai amănunţite sau se schimbă tipul listei, noile informaţii sînt afişate într-o nouă fereastră de tip **“browse”**.

(.) Replace curent – adică noile informaţii vor înlocui informaţiile din fereastra actuală.

Grupa cu titlul **“Prefered pane”** conţine următoarele butoane radio **:**

(.) Scope – dacă pentru o categorie de simboluri este posibilă afişarea mai multor tipuri de ferestre **“browse”**,atunci tipul implicit de fereastră, care se afişează prima dată este de tip **S (Scope)** adică domeniu.

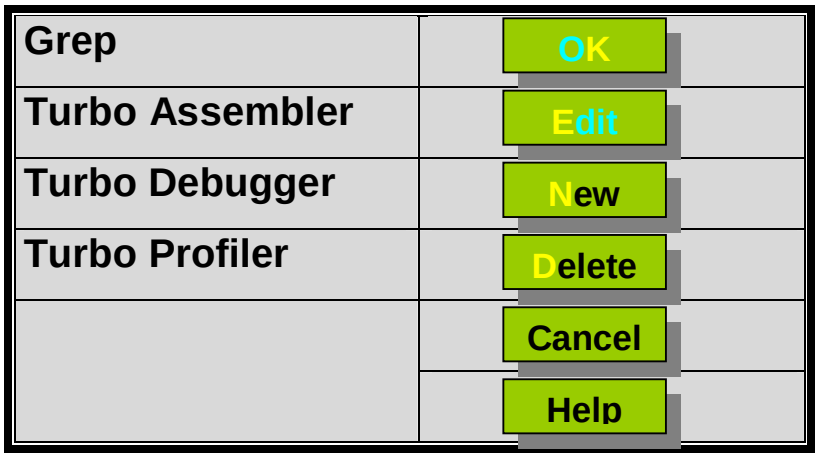
(.) Reference – tipul ferestrei este **R** adică referinţă.

Grupa cu titlul **“Display”** stabileşte modul de afişare a informaţiilor. Există următorii comutatori **:**

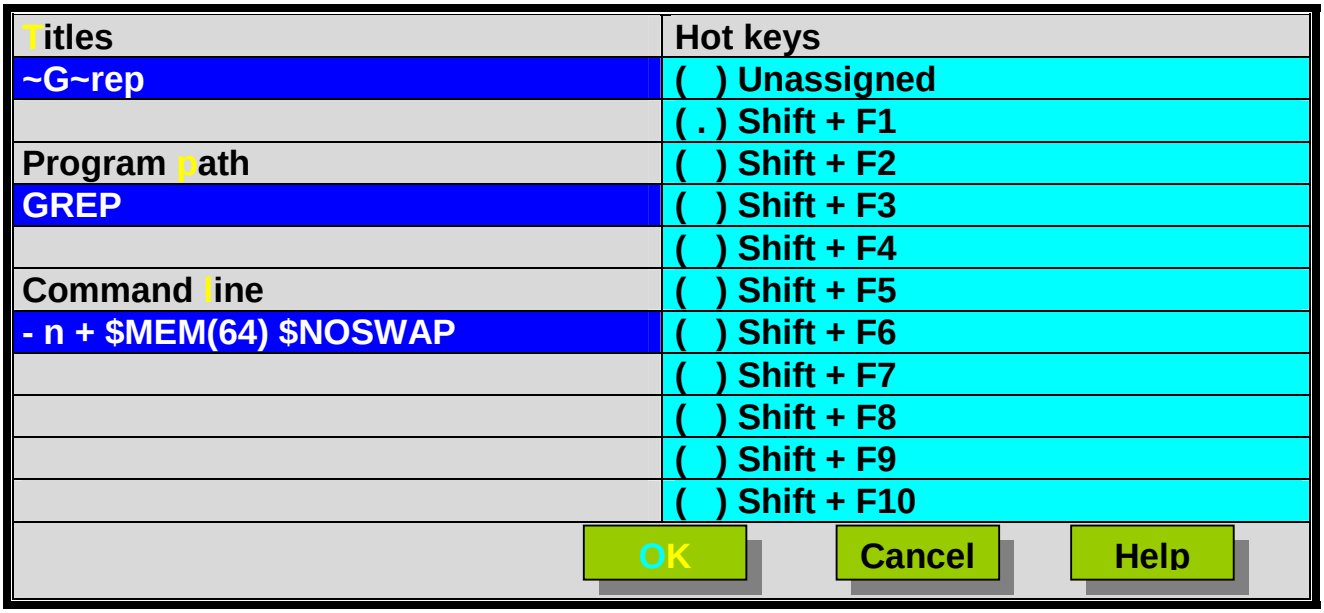
[X] Qualified symbols – dacă comutatorul este poziţionat, se afişează şi originea simbolului. De exemplu, metoda **"rotate"** a obiectului **"TObject1"** va fi afişată în lista de tip **S (Scope)** sub forma calificată **"Tobject1.rotate"**, dar dacă comutatorul nu este poziţionat, se afişează doar **"rotate"**.

[ ] Sort always – dacă comutatorul este poziţionat, elementele listei **"browse"** sînt afişate în ordine alfabetică, iar dacă nu este poziţionat ele sînt afişate în ordinea declaraţiilor.

Tools – permite introducerea utilitarelor noi în meniul **"Tools"** sau ştergerea ori modificarea celor vechi. Meniul poate să conţină cel mult zece programe. Utilitarele actuale sînt afişate într-o listă cu inscripaţia **"Program titles"** într-o fereastră de dialog cu următoarea formă **:**



Parametrii utilitarului actual al listei se modifică prin acţionarea butonului de comandă **"Edit"**. În urma apelării se afişează următoarea fereastră **:**



Dacă se doreşte includerea unui utilitar nou în lista de comenzi a meniului **"Tools"** atunci se va acţiona butonul de comandă **"New"**, care va afişa fereastra de mai sus. Comanda **"Delete"** permite ştergerea elementului actual din lista de utilitare apelabile. Cîmpul **"Title"** stabileşte sau modifică titlul comenzii

care este afişat în meniul **"Tools"**. Caracterul evidenţiat al comenzii trebuie încadrat între două caractere **"~"** (Tilda). Cîmpul **"Program path"** defineşte numele fişierului, care poate fi precedat de unitatea de disc şi de cale, în conformitate cu convenţiile folosite în sistemul **MS-DOS**. Cîmpul **Command line"** permite stabilirea parametrilor utilizaţi în linia de comandă, dacă este cazul. Grupa de butoane radio cu titlul **"Hot keys"** permite asocierea fiecărui utilitar la cîte o combinaţie funcţională din domeniul **"Shift + F2, …, Shift + F10"**.Dacă se alege varianta **"Unassigned"**, atunci utilitarului nu I se asociază nici o combinaţie de taste.

Environment este o comandă care afişează o listă de subcomenzi.

Preferences
Editor
Mouse
Startup
Colors

Subcomanda **"Preferences"** permite stabilirea diferitelor preferinţe legate de utilizarea mediului de programare.

Screen size	Source tracking
(.) 25 lines	(.) New window
( ) 43/50 lines	( ) Current window
Desktop file options	**Desktop file**
( ) Desktop only	( ) Current directory
(.) Desktop and symbols	(.) Config file directory
Auto save	**Options**
[ ] Editor files	[ ] Auto track source
[ ] Environment	[X] Close on go to source
[ ] Desktop	[ ] Change dir on open
OK	Cancel Help

Butoanele radio din grupa **"Screen sizes"** stabilesc dimensiunea ecranului. Se poate alege una din variantele **:** 25 linii pe ecran sau 43/50 linii pe ecran (43 la EGA şi 50 la VGA).

Butoanele radio din grupa **"Source tracking"** stabilesc modul de deschidere al fişierelor care se editează în mediu. Dacă se alege varianta **"New window"**, atunci fişierul deschis pentru editare este plasat într-o fereastră nouă de editare. Dacă se alege varianta **"Current window"**, noul fişier este plasat în fereastra actuală de editare.

Butoanele radio din grupa **"Desktop"** stabilesc directorul în care este salvat fişierul desktop **"TURBO.DSK"** sau fişierul **"BP.DSK"**. Fişierul **" *.DSK"** conţine informaţii referitoare la zona de lucru a mediului **:** listele cu istoricul, locaţiile punctelor de întrerupere, starea zonei de lucru. Dacă se alege varianta

“Current directory” , atunci fişierul **“ .DSK” va fi salvat în directorul curent. Dacă se alege varianta “Config file directory”,** atunci fişierul **“ *.DSK”** va fi salvat în directorul în care este salvat şi fişierul de configurare **“TURBO.TP”** sau **“BP.TP”**. Fişierul de configurare **“ *.TP”** conţine opţiunile actuale de compilare şi tabela cu comenzile editorului.

Butoanele radio din grupa **“Desktop file”** sînt utilizabile numai la compilatorul **“BP”**. Ele permit selectarea următoarelor opţiuni ale fişierului **“ *.DSK” :** dacă se alege varianta **“Desktop and symbols”**, atunci fişierul desktop va conţine şi informaţiile referitoare la simbolurile utilizate în program, iar dacă se alege varianta **“Desktop only”**, informaţiile referitoare la simboluri, nu sînt memorate în fişierul desktop.

Grupa cu titlul **“Auto save”** conţine trei comutatori **:**

[ ] Editor files

Dacă este cuplat, fişierele din ferestrele de editare încă nesalvate, sînt salvate automat în momentul părăsirii definitive sau temporare a mediului, la lansarea în execuţie a programului curent sau la declanşarea unei acţiuni de depanare.

[ ] Environment

Dacă este cuplat, în momentele mai sus menţionate, opţiunile actuale ale mediului, adică opţiunile de compilare şi tabela cu comenzile editorului, sînt salvate automat în fişierul de configurare **“ *.TP”**.

[ ] Desktop

Dacă este cuplat, în momentele menţionate la primul comutator, vor fi salvate automat informaţiile referitoare la listele cu istoricul, locaţiile punctelor de întrerupere şi starea zonei de lucru, în fişierul desktop **“ *.DSK”.** Comutatorul poate fi cuplat doar atunci, cînd este cuplat şi comutatorul **Environment.**

Grupa cu titlul **“Options”** conţine trei comutatori **:**

[ ] Auto track source

În stare cuplată reglează modul de utilizare al comenzii **“Track source”** prezentă în meniurile locale **“Browse”** şi **“Messages”**. Linia referenţiată de elementul actual al ferestrei **“Browse”** sau **“Messages”** va fi selectată automat, este afişată cu o culoare distinctă, deci nu este necesară acţionarea **“barei de spaţiu”**.

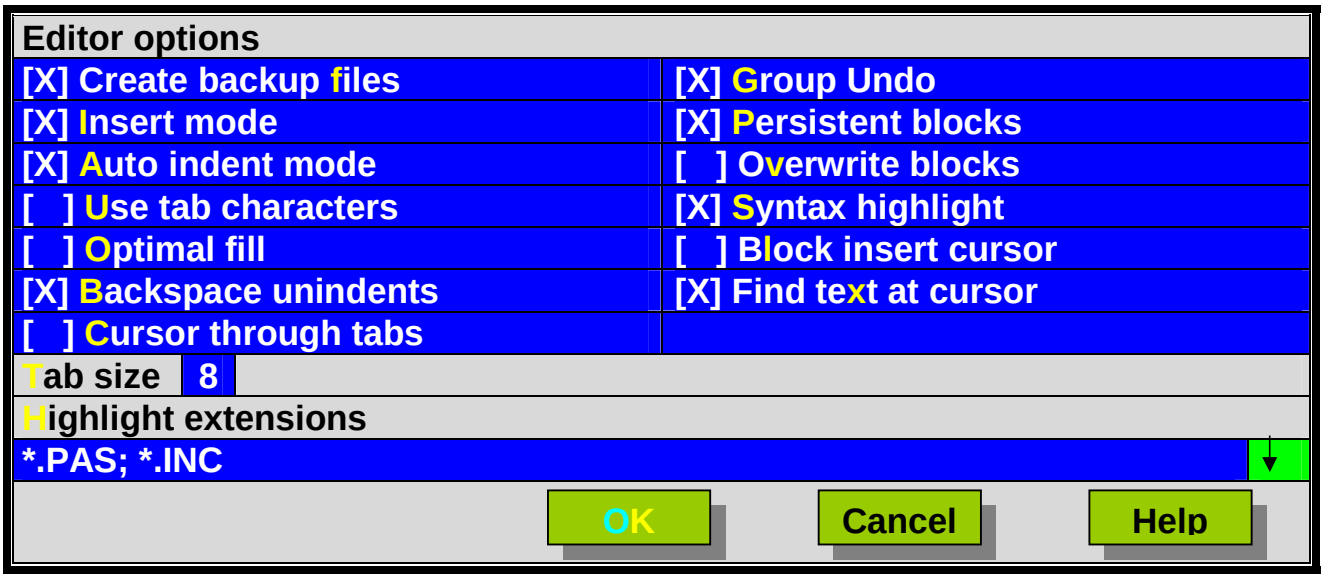
[X] Close on go to source

În stare cuplată reglează modul de utilizare al comenzii **“Goto source”** prezentă în meniurile locale **“Browse”** şi **“Messages”**. Cînd se trece la linia sursă a fişierului desemnat de elementul actual al ferestrei **“Browse”** sau **Messages”**, fereastra **“Browse”** sau **“Messages”** va rămîne deschisă.

[ ] Change dir on open

Dacă este cuplat, directorul fişierului deschis cu comanda **“Open”** din meniul **“Open”** va fi în continuare directorul actual iar dacă comutatorul nu este cuplat, atunci nu se va schimba directorul actual.

Subcomanda **“Editor”** permite parametrizarea editorului încorporat. Ea afişează următoarea fereastră de dialog **:**



[X] Create backup files

În stare cuplată, la comanda **"Save"** din meniul **"File"**, mediul va salva automat fişierul sursă original, cu extensia **".BAK"**, iar dacă nu este cuplat, fişierul sursă original nu este salvat.

[X] Insert mode (Ins, Ctrl + V)

În stare cuplată editorul lucrează în modul inserare, iar forma cursorului este o liniuţă, iar dacă nu este cuplat, forma cursorului este un dreptunghi.

[X] Autoindent mode (Ctrl + O, I)

În stare cuplată, prin acţionarea tastei **"Enter"**, cursorul va fi poziţionat în coloana primului caracter diferit de spaţiu din linia precedentă şi care nu este o linie numaicu blancuri. Acest lucru contribuie la claritatea programului.

[ ] Use tab characters (Ctrl + O, T)

În stare cuplată editorul inserează efectiv un caracter **"Tab"** (Cod ASCII 9) în momentul în care se tastează **"Tab"**. În stare decuplată, caracterul **"Tab"** este înlocuit cu spaţii. Numărul spaţiilor inserate este definit de cîmpul **"Tab size"**.

[ ] Optimal fill (Ctrl + O, F)

În stare cuplată editorul va încerca să folosească un număr minim de caractere la începutul fiecărei linii, cînd **"Auto indent mode"** este în stare cuplată.

[X] Backspace unindents (Ctrl + O, U)

În stare cuplată, dacă cursorul este pe o linie cu spaţii sau pe primul caracter diferit de spaţiu dintr-o linie, la acţionarea tastei **"Backspace"** toată linia se va alinia astfel încît să înceapă în aceiaşi coloană cu cea precedentă.

[ ] Cursor through tabs (Ctrl + O, R)

În stare cuplată cursorul se mişcă uniform cînd se ajunge la un caracter **"Tab"**, adică caracterul **"Tab"** este ignorat, altfel cursorul se deplasează înainte cu un număr de coloane care este specificat în cîmpul **"Tab size"**

[X] Group Undo

În stare cuplată comanda **"Undo"** din meniul **"Edit"** va anula ultimele 10 acţiuni, într-un singur pas. Pot fi grupate următoarele acţiuni **:** inserare, ştergere şi

suprascriere de text şi acţiunile de deplasare a cursorului. Dacă este decuplat, comanda **"Undo"** va restabili doar o singură acţiune.

[X] Persistent blocks

În stare cuplată marcarea (supraluminarea) de bloc va rămîne valabilă şi atunci, cînd se părăseşte domeniul blocului respectiv. În caz contrar, în momentul părăsirii blocului, marcarea blocului se termină.

[ ] Overwrite blocks

În stare cuplată şi dacă şi comutatorul **"Persistent blocks"** este activ, blocul marcat poate fi suprascris. În caz contrar, la prima modificare în interiorul blocului, marcarea este anulată.

[X] Syntax highlight

În stare cuplată diferitele construcţii ale limbajului Turbo Pascal, sînt colorate diferit.

[ ] Block insert cursor

În stare stare decuplată cursorul corespunzător modului de inserare este o liniuţă, iar al modului de suprascriere este un dreptunghi. În stare cuplată este invers.

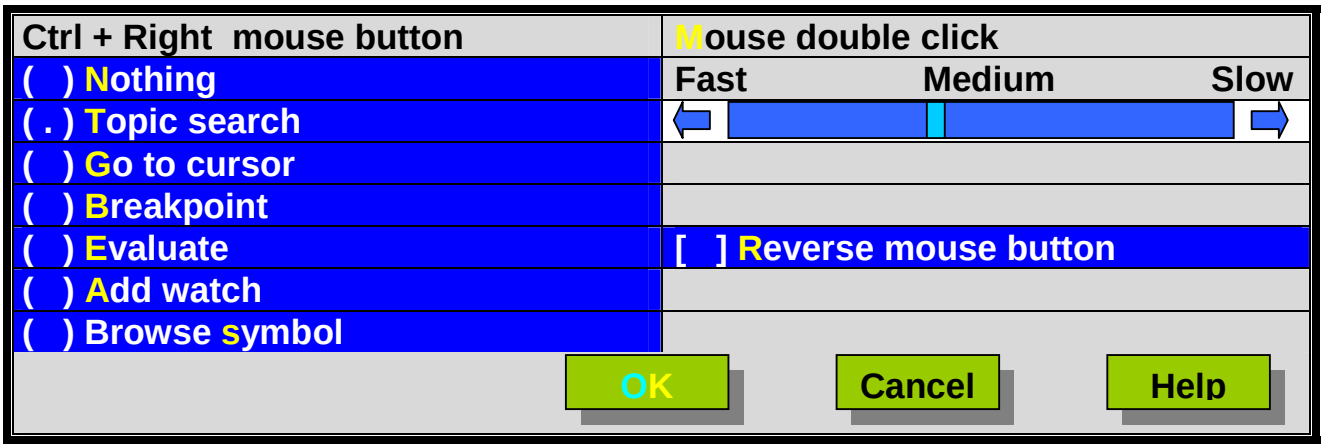
[X] Fint text at cursor

În stare cuplată, procesul de căutare este pornit de la poziţia actuală a cursorului, în caz contrar se porneşte de la începutul fişierului.

Cîmpul cu inscripţia **"Tab size"** permite definirea numărului de spaţii introduse cînd este acţionată tasta **"Tab"**. Valorile permise sînt înre 2 şi 16, iar valoarea implicită este 8.

Cîmpul cu inscripţia **"Highlight extension"** permite selectarea acelor extensii de fişiere, pentru care vor fi aplicate serviciile de colorare distinctă a diferitelor porţiuni din fişierul sursă, validate de comutatorul **"Syntax highlight"**. Colorările distincte vor fi aplicate implicit fişierelor cu extensia **".PAS; .INC"**.

Subcomanda **"Mouse"** afişează o fereastră de dialog care este destinată gestionării activităţii cu mouse-ul. Ea are forma următoare **:**



Grupa butoanelor **"Ctrl + Right mouse button"** ataşează o comandă la acţionarea butonului din dreapta al mouse-ului. Pot fi selectate următoarele variante **:**

( ) Nothing – butonul din dreapta este inactiv.

(.) Topic search – comanda **“Topic search” (Ctrl + F1)** din meniul **“Help”.**

( ) Go to cursor – comanda **“Go to cursor” (F4)** din meniul **“Run”**

( ) Breakpoint – comanda **“Add breakpoint (Ctrl + F8)** din meniul **“Debug”**.

( ) Evaluate – comanda **“Evaluate/Modify (Ctrl + F4)** din meniul **“Debug”**

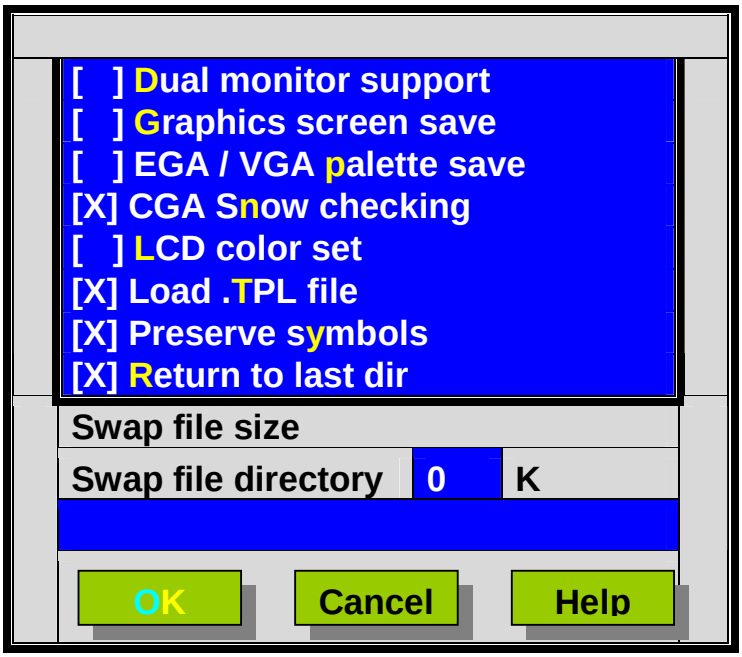
( ) Add watch – comanda **“Add watch” (Ctrl + F7)** din meniul **“Debug”.**

( ) Browse symbol – comanda **“Symbol”** din meniul **“Search”**, numai în mod protejat.

Timpul dintre două apăsări consecutive a butonului de mouse, pentru ca acţiunea să fie interpretată ca şi cum ar fi o apăsare dublă, poate fi stabilită în cîmpul cu titlul **“Mouse double click”**. Viteza creşte de la dreapta spre stînga.

Comutatorul **“Reverse mouse buttons”** se poziţionează atunci cînd se doreşte schimbarea rolului butonului din dreapta cu cel din stînga.

Subcomanda **Startup** afişează o fereastră de dialog în care pot fi precizaţi comutatorii care definesc modul de lansare al mediului de programare integrat.



[ ] Dual monitor support

Acest comutator se cuplează numai dacă computerul utilizează două plăci video care pot funcţiona în paralel. În acest caz, în urma cuplării opţiunii, pe monitorul central se afişează mediul de programare, iar pe celălalt se afişează datele de ieşire ale programului.

[ ] Graphics screen save

În stare cuplată, în modul de depanare, salvează întreaga memorie grafică. Mediul rezervă o memorie tampon de 8Ko, în momoria EMS, dacă această categorie de memorie este accesibilă, la sistemele pe EGA, VGA sau MCGA.

[ ] EGA / VGA palette save

În stare cuplată, în caz de depanare, salvează sau restabileşte paleta EGA / VGA. Dacă programul nu modifică regiştrii de paletă, nu este necesară cuplarea acestui comutator.

[X] CGA snow checking

În stare cuplată determină testarea şi anularea fenomenului deranjant de "ninsoare" a plăcilor CGA.

[ ] LCD color set

În stare poziţionată permite utilizarea mediului de programare şi pe ecranele calculatoarelor laptop.

[X] Load .TPL file

În stare cuplată, în momentul încărcării mediului, încarcă şi fişierul **"BP.TPL"** care conţine **unit-urile standard (System, Crt, Dos, Overlay, Printer, Strings)**.

[X] Preserve symbols

În stare cuplată **"reaminteşte"** informaţiile referitoare la simbolurile utilizate în ferestrele de tip **"browse"**. Dacă în urma unei compilări s-au găsit erori, încă pot fi folosite informaţiile din ultima compilare reuşită.

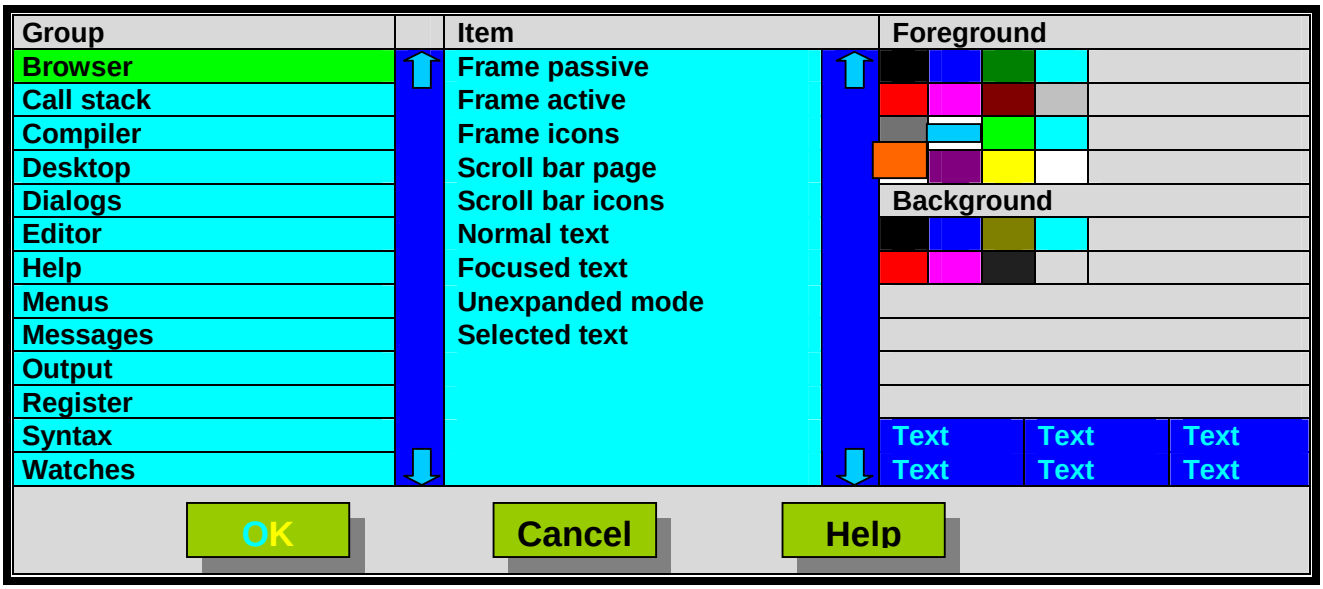
[X] Return to last dir

Comutatorul determină conservarea directorului curent. Dacă opţiunea este cuplată, la repornirea calculatorului, directorul curent va fi automat cel care a fost folosit în momentul ultimei ieşiri din mediul de programare.

"Swap file size" defineşte dimensiunea zonei de memorie care este rezervată pentru memorarea temporară a fişierului în curs de editare sau a zonei de lucru a editorului.

"Swap file directory" permite specificarea directorului în care este memorat fişierul de manevră al editorului. Numele acestui fişier începe cu **TP** şi are extensia **$$$**. Acest fişier este temporar, iar în momentul ieşirii din mediu este şters automat.

Subcomanda **"Colors"** afişează o fereastră de dialog care permite modificarea culorii diferitelor componente ale mediului.



Fereastra care se recolorează se selectează din coloana **"Group"**. Componenta de fereastră se alege din coloana **"Item"**. Culoarea caracterelor se alege din grupa culorilor **"Foreground"** iar culoarea fondului se alege din grupa culorilor **"Background"**.

Open – afişează o fereastră de dialog prin care se poate preciza numele unui fişier de configurare, salvat anterior cu comanda **"Save"** sau **"Save as"**. După precizarea numelui care de obicei este **"C:\BP\BIN\BP.TP"**, fişierul de configurare, în urma apăsării butonului**"OK"**, este încărcat.

Save – salvează opţiunile actuale de compilare şi tabela de comandă a editorului, în fişierul actual de configurare, care de obicei este **"C:\BP\BIN\BP.TP"**. Opţiunile referitoare la zona de lucru **"Desktop"** – liste cu istoricul,starea zonei de lucru, locaţiile punctelor de întrerupere sînt memorate în fişierul **"BP.DSK"**.

Save as – afişează o fereastră de dialog care permite introducerea numelui de configurare **"*.TP"**. În acest fişier, prin acţionarea butonului de comandă **"OK"**, vor fi salvate opţiunile actuale de compilare şi tabela de comandă a editorului. Fişierul salvat, poate fi încărcat ulterior cu comanda **"Open"**.

Meniul Window

Comenzile meniului **"Window"** permit deschiderea, aranjarea şi listarea diferitelor ferestre.

Tile – aşează diferitele ferestre deschise, una sub alta, astfel că toate ferestrele sînt simultan vizibile. Dacă numărul de ferestre deschise simultan este prea mare se emite semnalul de eroare **"Tile / Cascade area too small to complete reguest"**.

Cascade – permite aşeazarea ferestrelor deschise sub formă de grămadă. Prima fereastră ocupă întreaga zonă de lucru, celelalte sînt micşorate, fereastra activă este deasupra grămezii.

Close all – şterge zona de lucru, adică închide toate ferestrele deschise la un moment dat. Dacă există ferestre în care fişierul modificat nu a fost salvat, atunci se emit mesaje de avertisment.

Refresh display – restabileşte ecranul mediului. Această comandă se foloseşte atunci cînd ecranul mediului este suprascris de un program lansat în execuţie.

Size / Move(Ctrl + F5) – permite redimensionarea şi deplasarea ferestrei actuale. Redimensionarea ferestrei se realizează prin acţionarea simultană a tastelor **"Shift"** şi una din tastele cu săgeţi. Cînd se ajunge la dimensiunea dorită, se tastează **"Enter"**. Deplasarea poate fi realizată cu ajutorul tastelor cu săgeţi.După ce fereastra a ajuns la locul dorit, se tastează **"Enter"**.

Zoom(F5) – permite redimensionarea ferestrei active la dimensiunea maximă.Dacă fereastra are deja această dimensiune, comanda restabileşte dimensiunea la cea originală, adică comanda funcţionează ca un comutator.

Next(F6) – activează următoarea fereastră din ferestrele deschise la un moment dat.

Previous(Shift + F6) – activează fereastra precedentă, din ferestrele deschise la un moment dat.

Close(Alt + F3) – închide fereastra actuală. Dacă fişierul din fereastra actuală a suferit modificări şi încă nu a fost salvat, atunci mediul emite un mesaj de avertisment.

List(Alt + 0(zero)) – deschide o fereastră de dialog care listează toate ferestrele deschise în momentul respectiv.

Meniul Help

Comenzile meniului **"Help"** asigură accesul la diferitele componente ale documentaţiei în limba engleză, care însoţeşte mediul de programare.