1. Praca w środowisku Borland C/C++ 3.1.

1.1. Wiadomości wstępne.

Kod programu, napisany w języku C może być wykorzystywany w różnych środowiskach operacyjnych (UNIX, DOS, QNX, Linux, Windows), ale za każdym razem musi być ponownie kompilowany. Pod warunkiem ponownej kompilacji kod napisany w C może także poprawnie działać na różnych platformach sprzętowych (Intel 80x86, Pentium, RISC, SPARC, Alpha, itp.), także w systemach wieloprocesorowych i środowiskach rozproszonych. Instytut ANSI (Amerykański Narodowy Instytut Standaryzacji) wprowadził kilka wersji standardów dla języka C. Kompilatory różnych producentów (np. Borland, Microsoft, Watcom) są zazwyczaj zgodne ze standardem ANSI C. Wybór kompilatora C powoduje uznanie pewnej specyfiki. Dla przykładu, grafikę w środowisku DOS (BGI) pozwala realizować kompilator Borlanda (obecnie firma Inprise) tj.np. Borland C lub Turbo C. Nie można tych samych kodów skompilować z sukcesem np. przy pomocy Microsoft C/C++). *(„Programowanie w języku C .(Miniprzewodnik).", A Majczak).*

Firma Borland oferowała kompilator Turbo C, który został później „wchłonięty" i stał się częścią kompilatora C++. W wielu firmach nadal stosowane są bardzo udane wersje Borland C++ 3.0, 3.1, oraz Turbo C++. Jako kompilatory 16-bitowe, mogą wprawdzie być użyte do tworzenia aplikacji dla środowiska 32 bitowego, ale ich właściwym przeznaczeniem jest praca w środowisku DOS. Kompilator języka C w pakiecie Borland C++ 3.1 występuje obok kompilatora języka C++. Oba te kompilatory są instalowane łącznie (stanowią jeden plik wykonywalny). Do wersji 3.1 włącznie – Borland oferował dwa odrębne kompilatory – dla DOS i dla Windows. W wersjach 4.0 i następnych kompilator jest wspólny, wybiera się natomiast docelowe środowisko pracy programu – nazywane Target lub Platform.

Kompilatory różnych języków programowania rozpoznają rodzaj pliku dyskowego po rozszerzeniu: *.c - język C, *.cpp - C++, *.asm - Asembler. Aby kompilować i uruchamiać programy pisane w C, w kompilatorach Borland/Turbo C/C++ wystarczy zmienić domyślne rozszerzenie *.CPP na *.C. Według domyślnej konfiguracji: Options|Compiler|C++ Compiler| Use C++ Compiler| CPP Extension (only), co oznacza – stosuj kompilator C++ tylko przy rozszerzeniu *.CPP. Zapisanie pliku z własnym kodem w C (File|Save As... w postaci pliku *.C- zupełnie wystarczy.
Funkcje biblioteczne C pracujące w trybie tekstowym i słowa kluczowe C pisze się małymi literami. Dla języka C printf i PRINTF to dwa różne słowa.

1.2. Edytor Borland C + + 3.1.

Edytor programisty w IDE C++ jest edytorem wielo okienkowym. Można na ekranie otworzyć jednocześnie do 100 okien. Aby kompilator C nie miał wątpliwości, w którym okienku chcemy pracować, tylko jedno z tych okienek może być w danej chwili oknem aktywnym. Okno aktywne wyróżnia się na ekranie w środowisku DOS – podwójną ramką a w Windows – ciemnym paskiem tytułowym (pozostałe okna mają ramkę pojedynczą lub pasek jasny). *(HELP programu Borland C++ 3.1).*

1.2.1. Obsługa okien.

Jest dużo sposobów dostania się do otwartych, edytowanych okien, aby stały się aktywne:
- Kliknąć na oknie.
- Nacisnąć Alt+# (gdzie # jest numerem okna).
- Wybrać w oknie z wykazem (listą) otwartych okien.

- Naciskając F6 przechodzi się przez otwarte okna.
- Aby zamknąć aktywne okno edycji, wybieramy Window¦Close.

W menu Window znajduje się wiele opcji dotyczących nie tylko okien edycji. Oprócz wybrania odpowiedniej pozycji z tego menu, można także obsługiwać okna przy pomocy myszy, bądź klawiatury.

Mysz:
- można ciągnąć okno po ekranie (za pasek tytułowy podobnie jak w Windows),
- zmieniać wymiary okna (Mowe/Size) ciągnąc za ramkę,
- przewijać tekst w oknie posługując się paskami przewijania.

Klawiatura:
- skróty klawiszowe podane w menu,
- klawisze kursora (ze strzałkami) powodują po wydaniu rozkazu Size/Move przesuwanie okna,
- [Shift]+[→] powoduje zmianę wymiarów okna.

1.2.2. Redagowanie tekstu.

- Aby zakończyć linie, naciskamy Enter.
- Aby zakończyć redagowanie, naciskamy F10.
- Ponadto do dyspozycji mamy polecenia menu Edit (Edycji), opisane w dalszej części.

Można wybierać (podświetlać) tekst z klawiatury lub myszą:
- Klawiatura: Przy jednoczesnym naciśnięciu klawisza Shift, zaznaczamy obszar z tekstem za pomocą klawiszy kursora.
- Mysz: Naciskając lewy przycisk myszy, przesuwamy wskaźnikiem myszy ponad tekstem do zaznaczenia, linia po linii, gdy wskaźnik myszy przeciągniemy poza okno to obszar tekstu w oknie zacznie się przesuwać ukazując niedostępny do tej pory tekst z dalszej części pliku, gdy lewy klawisz zostanie puszczony, zaznaczanie tekstu zostanie zakończone. Aby zaznaczyć pojedynczą instrukcję, bądź łańcuch znaków wystarczy na nich kliknąć dwukrotnie. Po wybraniu tekstu, rozkazy w menu edycji stają się dostępne.

Maksymalna szerokość(długość) linii w oknie edycji wynosi 1023 litery. Komputer będzie sygnalizował sygnałem dźwiękowym, jeżeli napisane zostanie więcej liter. Maksymalna liczba linii w pliku wynosi 65,534. Możliwe jest w przybliżeniu do 6 Mb miejsca na otwarte pliki, oraz bloki do kopiowania, wycinania i wklejania.

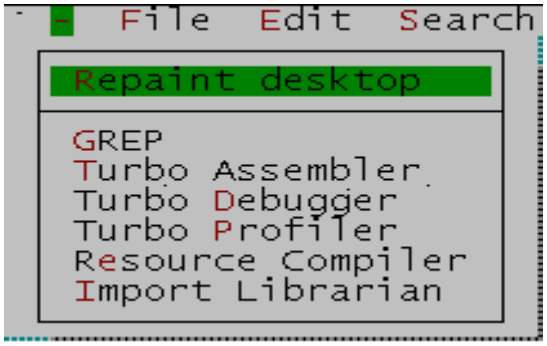
1.3. Borland C + + , opis poleceń menu głównego.

Aby wejść do menu głównego programu: naciskamy F10, Alt+[wyróżniona litera w opcji] lub klikając gdziekolwiek na belce, można wybrać dowolny z rozkazów menu:

- (System)
File (Plik)
Edit (Edycja)
Search (Poszukiwanie)
Run (Prowadź, uruchom)
Compile (Kompiluj)
Debug (Usuwaj błędy z programu)
Project (Projekt)
Options (Opcje)
Window (Okno)
Help (Pomoc)

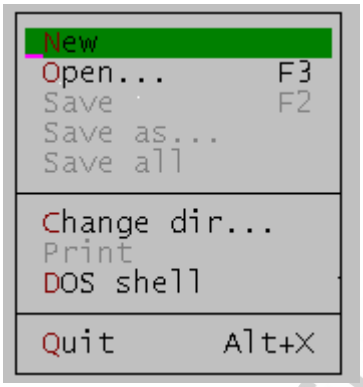
1.3.1. - (System).

Po rozwinięciu - menu, widoczne jest oddzielone polecenie Repaint desktop, oraz pod kreską wykaz programów, które zostały zainstalowane z Options¦Transfer.



Repaint desktop. Rozkaz ponownie przerysuje ekran. Stosowane, jeżeli rezydentny program zostawił zabłąkane litery na ekranie.
Transfer Programs. Dowolne programy zainstalowane z okna dialogu przenoszenia(Transfer) ukazują się tutaj.

1.3.2. File (Plik).



New (Nowy)
Open (Otwórz)
Save (Zapisz)
Save as (Zapisz jako)
Save all (Zapisz wszystko)
Change dir (Zmień katalog)
Print (Drukuj)
DOS shell (Przejdź (chwilowo) do DOS)
Quit (Opuść).

File¦New (Nowy).
Otwiera nowe okno edycji o tymczasowej nazwie NONAMExx.Cpp (xx oznacza liczbę od 00 do 31) i automatycznie tworzy nowe aktywne okno edycyjne. Te bez nazwy (Noname) pliki są używane jako tymczasowy bufor edycji. Borland C + + skłania do zastąpienia Noname, nazwą pliku, kiedy będzie zapisywany.
File¦Open (Otwórz).
Otwiera wskazany (istniejący już) plik z listy i umieszcza jego zawartość w nowo otwartym oknie edycji.
File¦Save (Zapisz).
Zapisuje zawartość aktywnego okna edycji. Opcja ta jest nie aktywna, gdy plik nie ma określonej nazwy (tzn. NONAME), lub okno jest nieaktywne. Zapisuje pod bieżącą nazwą, na bieżącym dysku, w bieżącym katalogu.
File¦Save as (Zapisz jako)...
Używa się tej opcji nadając nazwę plikowi NONAME, zapisując tą zawartość pod inną nazwą lub w innym katalogu.
File¦Save all (Zapisz wszystko).
Zapisuje modyfikacje we wszystkich aktywnych oknach.
File¦Change dir (Zmień katalog)...
Zmiana aktualnej ścieżki dostępu. Aktualna ścieżka dostępu jest jedna gdzie Borland C + + zapisuje pliki i szuka plików. Kiedy używamy względnych ścieżek w Options¦ Directories dialog box, są one względne tylko do tej aktualnej ścieżki dostępu.
File¦Print (Drukuj).
Drukuje zawartość aktywnego okna: Edit, Output, lub Message.

File¦DOS shell (Przejdź (chwilowo) do DOS).
Można opuszczać Borland C++ chwilowo wykonując rozkazy lub wchodzić do innych programów. Aby wrócić do Borland C++, piszemy w linii poleceń słowo Exit i naciskamy Enter. Można też używać przenoszenia z pozycji w - (System) menu szybko przełączając się do innych programów bez opuszczania Borland C++.
File¦Quit (Opuść).
Wyjście z Borland C++ (usuwa z pamięci) i wraca do DOS.

1.3.3. Edit (Edytuj).

Menu edycji dostarcza rozkazy wycinania (cut), kopiowania (copy) i wklejania tekstu (paste) w edytowanym (aktywnym) oknie . Można też anulować zmiany i odwrotność zmiany, które właśnie zostały anulowane. Można otwierać (Clipboard-schowek) okno podręcznego notatnika oglądać lub redagować jego zawartości i kopiować do niego tekst z okien Message, Output, oraz Help.

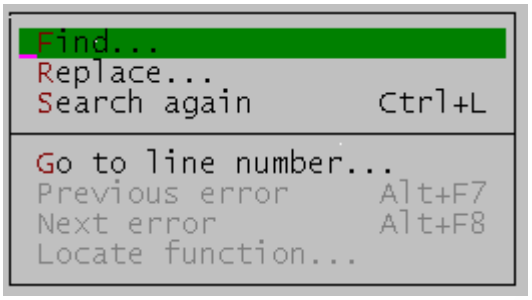
Undo (Anuluj, cofnij)
Redo (Ponownie wykonaj)
Cut (Wytnij)
Copy (Kopiuj)
Paste (Wklej)
Clear (Czyść, usuń)
Copy example (Kopia przykładu)
Show clipboard (Podręczny notatnik, schowek)

Edit¦Undo (Anuluj, cofnij).
Anuluje ostatnio wydany rozkaz lub wykonaną czynność.
- Anuluje nie może odwracać jakiegokolwiek przełącza układu, który ma globalny skutek; na przykład, Ins / Ovr.
- Anuluje zmianę tylko na ostatniej modyfikowanej albo usuwanej linii.
Edit¦Redo (Ponownie wykonaj).
Ponownie wykonuje rozkaz odwraca skutek ostatniego anulowania. Seria Redo (ponownie wykonaj) odwraca skutki serii Undo (Anuluj).
Edit¦Cut (Wytnij).
Rozkaz kopiuje wybierany tekst z dokumentu i umieszcza w podręcznym notatniku, jednocześnie usuwając go z dokumentu. Można wtedy wybierając Edit¦Paste wklejać ten tekst do dowolnego innego dokumentu (lub gdzie indziej w tym samym dokumencie). Można wklejać ten tekst wiele razy.
Edit¦Copy (Kopiuj).
Kopiuje zaznaczony tekst do podręcznego notatnika, aby wkleić go w innym miejscu. Można także kopiować tekst z okna HELP:
-klawiatura, przy naciśniętym Shift i klawiszami kursora (strzałki) wybieramy (podświetlając) tekst, który chcemy kopiować.
- myszą, klikając i przeciągając wybieramy tekst, który chcemy kopiować.
Edit¦Paste (Wklej).
Wstawia tekst z podręcznego schowka do aktualnego okna przy pozycji kursora. Tekst, który faktycznie jest wklejany jest aktualnie znaczony w podręcznym notatniku.
Edit¦Clear (Czyść, usuń).
Usuń tekst zaznaczony. Tekstu tego nie można ponownie wklejać, ponieważ nie jest on zapamiętywany w podręcznym schowku. Odzyskanie tego tekstu następuje poprzez opcje: anuluj poprzedni krok.

Edit¦Copy example (Kopia przykładu).
Kopiuje przykład z okna pomocy do podręcznego schowka, można go skopiować poleceniem wklej do okna edycji (aktywnego).
Edit¦Show clipboard (Podręczny notatnik,schowek).
Pokaż podręczny notatnik, schowek z tekstem skopiowanym lub wyciętym z innych okien.
Aktualnie wybierany (podświetlany) tekst jest przez Borland C++ używany gdy będzie wybierane Edit¦Paste .

1.3.4. Search (Przeszukaj).

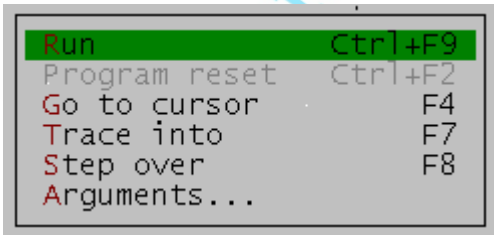
Menu poszukiwania dostarcza rozkazy badania tekstu, rozmieszczenia błędów w plikach, oraz poszukuje deklaracje funkcji.



Find... (Znajdź).
Replace... (Zastąp).
Search again (Przeszukaj ponownie).
Go to line number...(Idź do lini numer)
Previous error (Poprzedni błąd)
Next error (Następny błąd)
Locate function... (Lokalizuj funkcje).

Search¦Find (Znajdź)...
Piszemy, co w tekście chcemy szukać, jakiego słowa lub tekstu.
Search¦Replace (Zastąp)...
Piszemy, co w tekście chcemy szukać i na co chcemy to zastąpić.
Search¦Search again (Przeszukaj ponownie).
Rozkaz powtarza ostatnie Search¦Find albo Replace.
Search¦Go to line number (Idź do lini numer)...
Pyta o numer szukanej linii. Borland C++ przedstawia aktualny numer linii i kolumny.
Search¦Previous error (Poprzedni błąd).
Rozkaz przemieszcza kursor do poprzedniego błędu albo ostrzeżenia. Ten rozkaz jest dostępny tylko, jeżeli są wiadomości w Massage Window, które dołącza numer linii.
Search¦Next error (Następny błąd).
Rozkaz przemieszcza kursor do następnego błędu albo ostrzeżenia.
Search¦Locate function (Lokalizuj funkcje)...
Pokazuje okno dialogu Lokalizacja Funkcji, gdzie wpisywana jest nazwa szukanej funkcji. Ten rozkaz jest dostępny tylko podczas usuwania błędów z programu. Znajduje deklarację funkcji, nie przykład użycia funkcji.

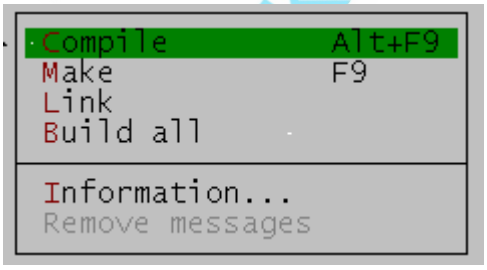
1.3.5. Run (Prowadź, uruchom).



Run (Prowadź, uruchom)
Program reset (Ponownie ustawia)
Go to cursor (Idzie do kursora)
Trace into (Do wskazanej)
Step over (Krok ponad)
Arguments (Argumenty)

Run¦Run (Prowadź, uruchom) Ctrl F9.
Polecenie RUN powoduje wykonanie programu, uwzględniając przy tym ewentualne parametry określone w okienku dialogowym polecenia Arguments, poprzedzone ewentualnie jego kompilacją.
Wykonanie programu zależy od ustalenia opcji-argumentów w podokienku parametrycznym okienka dialogowego polecenia Debugger. Jeżeli oba parametry są wyłączone, to wprawdzie śledzenie programu nie będzie możliwe, ale za to na wykonanie programu będzie wyznaczony większy obszar pamięci operacyjnej.
W przypadku, gdy parametr Integrated jest włączony (domyślnie) kod wynikowy programu będzie zawierał wszystkie informacje niezbędne w sesji debuggera. Przy tym, jeśli tekst źródłowy programu nie został zmodyfikowany od czasu jego ostatniej kompilacji, to program zostanie wykonany do najbliższego punktu przerwania, a gdy nie zostały one ustawione – do końca kodu.
Run¦Program reset (Ponownie ustawia).
Ładuje ponownie aktualny program z dysku. Zwalnia pamięć zajętą i przydzieloną przez program użytkownika i powoduje zamknięcie wszystkich zbiorów, które zostały otwarte w tym programie.
Jeśli debugger systemowy nie został zainicjowany, to polecenie jest wyłączone. Podczas sesji debuggera polecenie może być zainicjowane z dowolnego miejsca systemu przez naciśnięcie klawiszy Ctrl-F2.
Run|Go to cursor (Idzie do kursora) F4.
Polecenie to powoduje rozpoczęcie lub kontynuację wykonania programu od bieżącej pozycji wykonania, którą z chwilą rozpoczęcia sesji debuggera jest pierwszy wiersz programu, a w innym przypadku miejsce, w którym ostatnio debugger zatrzymywał wykonanie programu – do wiersza programu, w którym jest aktualnie umiejscowiony kursor lub do pierwszego napotkanego punktu przerwania.
Run¦Trace into F7 (Do wskazanej) i Run¦Step over F8 (Krok ponad).
Polecenia te powodują wykonanie jednej (bieżącej) instrukcji programu. Różnica w wykonaniu tych poleceń występuje w przypadku, gdy bieżącą instrukcją jest instrukcja wywołania funkcji. W przypadku polecenia Trace into nastąpi przejście do treści danej funkcji.
Natomiast, gdy zostanie wywołane polecenie Step over, to nastąpi jej wykonanie (w jednym kroku), po czym debugger ustali wiersz z następną instrukcją po instrukcji wywołania danej funkcji, jako aktualną pozycję wykonywania programu.
Run¦Arguments (Argumenty)...
Polecenie to pozwala dla wykonywanego programu określić DOS-owskie argumenty wykonania. Do ich wprowadzenia służy podokienko wejściowe okienka dialogowego, które ukazuje się na ekranie po zainicjowaniu tego polecenia.
W podokienku tym argumenty określa się w podobny sposób, jak w systemie operacyjnym DOS, pomija się jedynie nazwę programu.

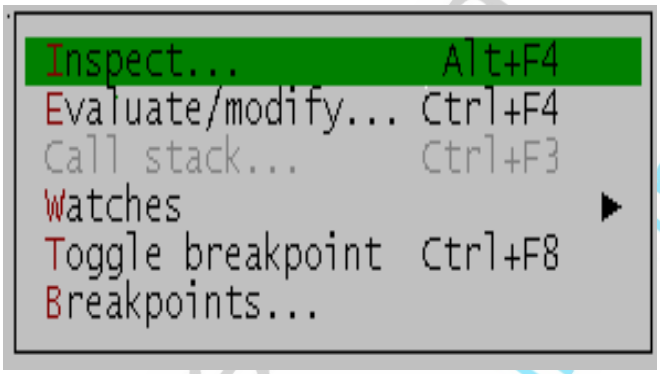
1.3.6. Compile (Kompiluj).



Compile (Kompiluj)
Make (Rób)
Link (Łączenie)
Build all (Konstrukcja wszystko)
Information (Informacja)
Remove messages (Usuwa wiadomości)

Compile¦Compile (Kompiluj) Alt F9.
Równoważne z linią poleceń (command-line) BCC: -c. Rozkaz kompiluje aktywny, edytowany plik (.C albo.Cpp) do pliku .Obj. Kiedy Borland C++ kompiluje, wyświetla okno stanu (status box) do pokazu przebiegu kompilacji będzie ono przedstawiać rezultat kompilowania: numer kompilowanej linii (lines compiled), numery poszczególnych błędów i ostrzeżeń, oraz ilość dostępnej pamięci (available memory). Gdyby zdarzały się jakiekolwiek błędy, okno to staje się aktywne i pokazuje (podświetlając) pierwszy występujący błąd.
Compile¦Make (Rób) F9.
Równoważne z linią poleceń (command-line) Bc (Ide): /m. Rozkaz ten wzywa Projekt Zarządce (Menager) do wykonania projektu celu, pliku wykonywalnego nazwa.exe .
1. Projekt plik (. Prj) wyszczególniany z Project¦Open Project command.
2. Nazwa pliku w aktywnym oknie edycji.
Gdy żaden projekt nie jest zdefiniowany, to otrzymywany projekt będzie nie definiowany przez plik Tcdef.dpr . Ponownie buduje projekt, ale tylko pliki, które nie są aktualne.
Compile¦Link (Łączenie).
Łączy (linkuje) biorąc pliki definiowane w aktualnym pliku projektu (do postaci plik.exe).
Compile¦Build all (Konstrukcja wszystko).
Równoważne z linią poleceń (command-line) Bc (Ide): / b. Rozkaz ten ponownie buduje wszystkie pliki w projekcie, nie zważając na to, czy są one aktualne.
Compile¦Information (Informacja).
Pokazuje okno z informacjami, statystyką o aktualnym pliku.
Compile¦Remove messages (Usuwa wiadomości).

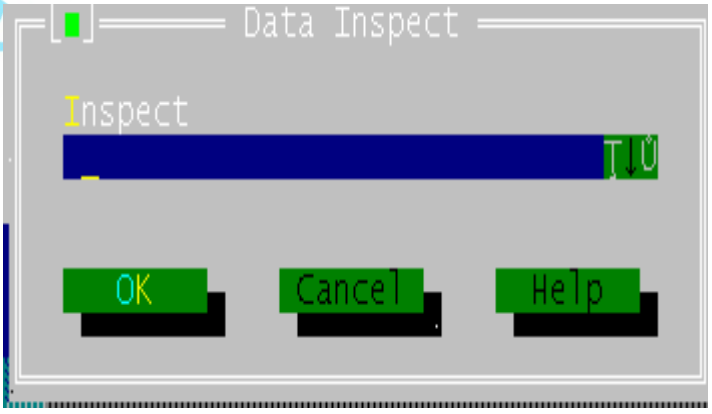
1.3.7. Debug (Usuwaj błędy z programu).



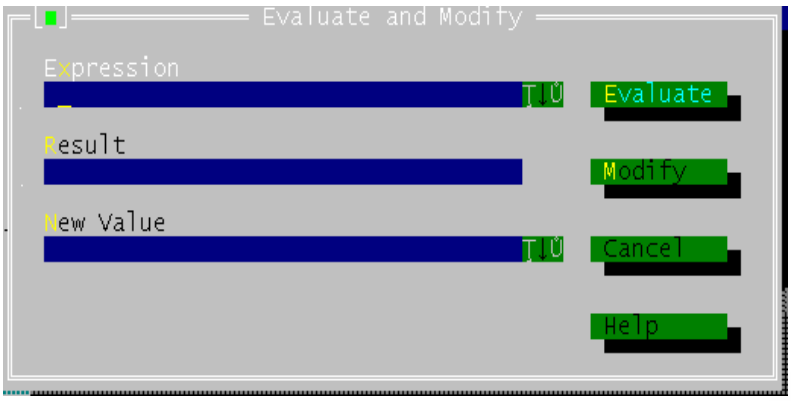
Inspect (Badaj)
Evaluate/modify (Oceniaj / modyfikuj)
Call stack (Połączenie ze stosem, odwołanie)
Watches (Obserwacja)
Toggle breakpoint (Przełączanie punktu przerwania)
Breakpoints (Punkty przerwania)

Debug¦Inspect (Badaj)...

Otwiera inspektora i pokazuje wartość zmiennej pod kursorem w oknie edycji. Jeżeli znaczymy wyrażenie używające INS, inspektor pokazuje wartość podświetlanego wyrażenia.

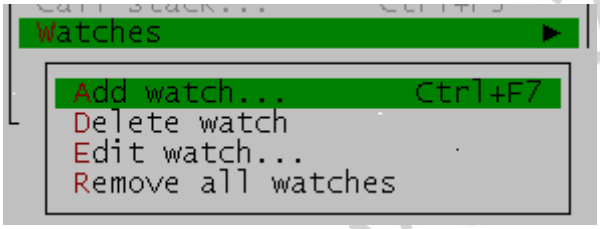


W Borland C + +, można badać następujące typy:
- prosty (simple), (porządkowy, char lub unsigned long)
- tablicowy (arrays)
- wskaźnikowy (pointers)
- structures, classes, types, unions, function.



Debug¦Evaluate / modify (Oceniaj / modyfikuj)...
Okno dialogowe Oceny i modyfikacji, tu możemy:
- oceniać zmienne albo wyrażenia
- przeglądać wartości dowolnej zmiennej
- zmieniać wartość danych

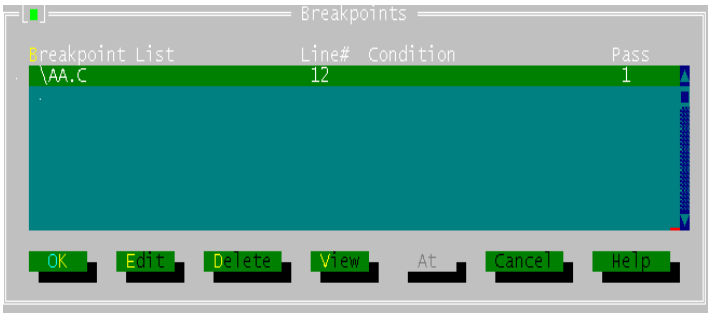
Debug¦Call stack (Połączenie ze stosem, odwołanie)...
Podczas sesji debuggera powoduje ono otwarcie specjalnego okienka, które zawiera listę odwołań do funkcji i metod, prowadzących do aktualnego miejsca wykonywania programu. Razem z nazwami są wyświetlane ich parametry.
Debug¦Watches (Obserwacja).
Polecenie to wyświetla poniższe opcje



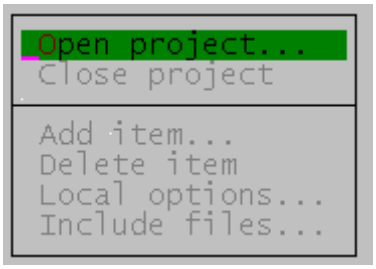
Add watch (Dodaj do obserwacji)
Delete watch(Usuwa z okna obserwacji)
Edit watch (Edytuje)
Remove all watches (Usuwa wszystkie obserwacje)

Opcjami tymi działamy na okno edycyjne Watches. I tak na przykład wprowadzamy do okna zmienne lub wyrażenia, które chcemy badać za pomocą Add watch.
Jeśli w oknie tekstowym programu, wskazujemy zmienną za pomocą migającego kursora, to ta zmienna zostaje domyślnie wskazana w pasku edycyjnym. Gdy w oknie Watches podświetlimy którąś ze zmiennych, to ona zostanie uwzględniona podczas polecenia Delete watch lub Edit watch.
Przez zainicjowanie polecenia RUN w okienku Watches, przy wszystkich wyspecyfikowanych zmiennych zostaną wyświetlone ich ostatnie wartości.
Debug¦Toggle breakpoint (Przełączanie punktu przerwania).
Umieszcza bezwarunkowy punkt przerwania na linii gdzie kursor jest ulokowany.

Debug¦Breakpoints (Punkty przerwania)...
Otwiera okno dialogu, gdzie wyszczególnione są czynniki o warunkowych i bezwarunkowych punktach przerwania.



1.3.8. Project (Projekt).



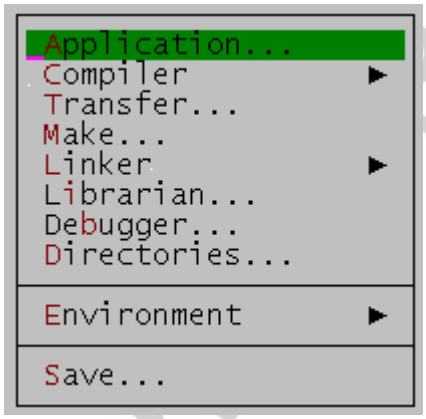
Open project... (Otwórz projekt).
Close project (Zamknij projekt).
Add item... (Dodaje pozycję).
Delete item (Usuwa pozycję).
Local options... (Lokalne opcje).
Include files... (Dołączaj pliki).

Project¦Local options (Lokalne opcje)...
- włącza opcje linii poleceń dla projektu pliku modułu
- daje wyraźną nazwę i ścieżkę dla pliku obiektu
- wybiera tłumacza dla modułu

Project¦Include files (Dołączaj pliki)...
Można oglądać, które pliki są włączane przez plik projektu do okna projektu.

1.3.9. Options (Opcje).

Najważniejsze z rozkazów w tym menu dotyczą: Kompilatora(Compiler), programu łączącego (Linker) i otoczenia(Environment).



Application (Aplikacje)
Compiler (Kompilator)
Transfer
Make (Rób)
Linker (Program łączący)
Librarian (Biblioteki)
Debuger
Directories (Katalogi)
Environment (Otoczenie)
Save (Zapisywanie)

Options¦Application (Aplikacje)...
Używa się tego okna dialogu, aby rozpocząć kompilację i łącząc (linkować) dla Windows lub DOS uzyskać program wynikowy .

Options¦Compiler (Kompilator).
Code Generation... (ustawienie modelu pamięci).
Advanced Code Generation... – dodatkowe opcje gen. kodu (np. liczby zmiennopozycyjne, dla jakiego procesora itp.)
Entry/Exit Code... (konwencje We/ Wy kodu, opcje stosu).
C++ Options... (Opcje kompilowania, rodzaje szablonów dla kompilatora).
Advanced C++ Options... (dodatkowe opcje, zgodność z poprzednimi wersjami, dziedziczenie, wskaźniki).
Optimizations... (Optymalizacja rozmiaru kodu i szybkości).
Source... (typ kodu źródłowego: słowa kluczowe Borland C++, ANSI, UNIX V, Kernighan and Ritchie, zagnieżdżenie komentarzy, liczba znaczących liter w identyfikatorze).
Messages (okna dialogu komunikatów o błędzie, kategorii błędów:
Display... Wyświetl.
Portability... Przenoś.
ANSI Violations... Ansi pogwałcenia.
C++ Warnings... C + + ostrzeganie.

Frequent Errors... Częste błędy.
Less Frequent Errors... Mniej częste błędy.)

Names... (można zmieniać segment domyślny, grupę i klasowe nazwy dla kodu, dane i BSS segmenty.)

Options¦Transfer ...

Można dodawać lub usuwać programy w menu System.

Options¦Make (Rób)...

Okna dialogu gdzie wybiera się warunki dla kompilacji projektu. Przerwanie wykonania, typy kompilowania, import bibliotek, kontrole błędów.

Options¦Linker (Program łączący)...

Te rozkazy prowadzą do okien dialogu gdzie dokonuje się wyborów, które dotyczą łączenia.
Settings... Umieszczania. Libraries... Biblioteki.

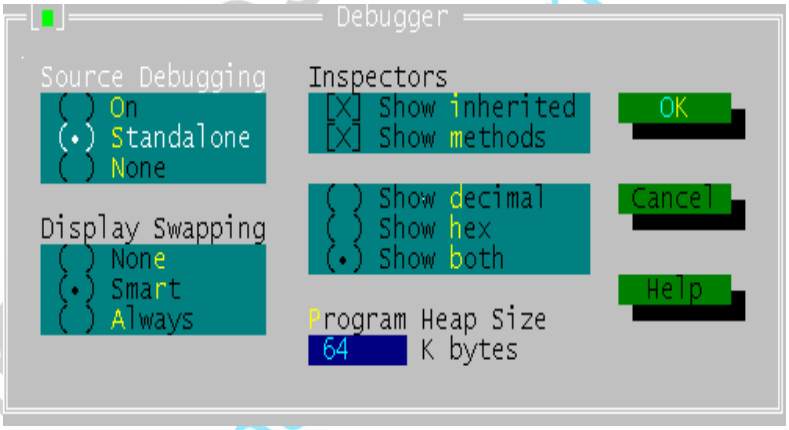
Options¦Librarian (Biblioteki)...

Wybieramy opcje, które dotyczą wbudowywanego bibliotekarza. Wbudowywany bibliotekarz łączy pliki .OBJ w projekcie do pliku .LIB.

Options¦Debugger...

Polecenie to służy do ustalenia parametrów dotyczących pracy debuggera systemowego.

Source Debugging: On , Standalone , None.



Pozwala określić, czy informacje przeznaczone dla debuggera mają być dołączone do zbioru wynikowego .EXE i jak program (po skompilowaniu) ma być wykonany w ramach zintegrowanego systemu programowania.

Po włączeniu parametru Standalone, wszystkie informacje, niezbędne dla debuggera zewnętrznego, będą dołączone do utworzonego podczas kompilacji zbioru .EXE.

Display Swapping: None, Smart, Always

Za pomocą jednej z tych opcji określa się sposób zmiany ekranu podczas sesji debuggera.

None – brak zmiany ekranu (opcja ta może być stosowana w przypadku, gdy debugger nie wyświetla na ekranie żadnych informacji).

Smart – zmiana ekranu systemowego na ekran wykonawczy (wyjściowy) tylko podczas wykonywania instrukcji programu powodujących wyświetlanie danych na ekranie oraz instrukcji wywołania procedury, po czym powrót do ekranu systemowego (standardowo).

Always – zmiana ekranu systemowego na ekran wykonawczy podczas wykonywania instrukcji.

Inspectors

[X] Show Inherited - pokazuje dziedziczenie
[X] Show Methods - pokazuje metody

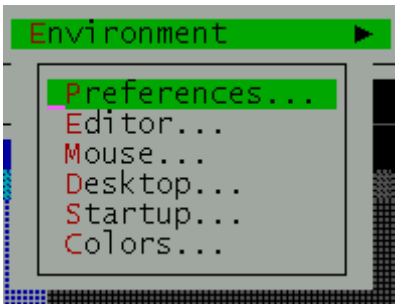
Te opcje mówią, aby debugger wyświetlał metody (kolejno), kiedy będzie badany obiekt.

( ) Show Decimal - dziesiętne
( ) Show Hex - szesnastkowo
(¨) Show Both - oba

Te opcje kontrolują jak debuger pokazuje wartości w oknach Inspektora.

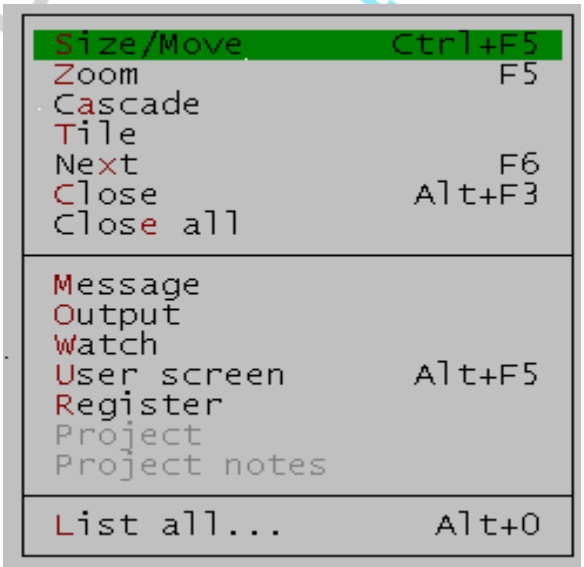
Program Heap Size – Okno rozmiaru stosu programu, wyszczególnia ile pamięci Borland C++ powinien wyznaczać, kiedy usuwane są błędy z programu.

Options¦Directories (Katalogi)...
Otwiera okno, w którym ustawiane są ścieżki do roboczych katalogów kompilatora.
Include Directories - katalog z dołączanymi plikami nagłówkowymi np. stdio.h itp.
Library Directories – to katalogi zawierające biblioteki statyczne i dynamiczne .LIB i .DLL.
Output Directory – to katalog wyjściowy, w którym będą umieszczane powstające pliki wynikowe .OBJ i .EXE. Domyślnie przyjmowany jest katalog źródłowy, w którym znajduje się plik *.C lub *.C++.
Source Directories – katalog gdzie połączony w jedną całość, debuger szuka kodu źródłowego do bibliotek, które nie należą do otwartego projektu.
Options¦Environment (Otoczenie)...
Opcja ta zawiera polecenia służące do ustalania parametrów pracy zintegrowanego systemu programowania.



Preferences (Preferencje)...
Editor (Edytor)...
Mouse (Mysz)...
Desktop (System)...
Startup (Uruchomienie)...
Colors (Kolory)...

Preferences (Preferencje)...
Screen sizes- wybór jednego z dwu trybów pracy ekranu systemowego: 25 wierszy i 43/50.
Source tracking- określa odpowiednie okienko edycji.
Auto-Save- Sterowanie automatycznym zapisem.
Editor (Edytor)...
Służy do ustalenia trybów pracy edytora, tabulacji, tworzenia plików bak, zachowania tekstu.
Mouse (Mysz)...
Ustalenie działania prawego klawisza myszy oraz szybkości reakcji na podwójne kliknięcie.
Desktop (System)... Stan systemu. Określa ustawienia dla schowka, punktów przerwań, śledzenia zmiennych, okna historii, oraz zamykania i otwierania okien.
Startup (Uruchomienie)...
Służy do określenia opcji uruchomienia zintegrowanego systemu programowania.
Colors (Kolory)...
Pozwala na zdefiniowanie kolorów tła, napisów i różnych elementów wszystkich okien, wyświetlanych na ekranie.
Options¦Save (Zapisywanie)...
Zapisuje zmiany, jakie dokonano w ustawieniach dostępnych przez Options. Borland C++, zapisuje (składa) opcje w trzech plikach:
- przystosowywane otoczenie (Tcconfig.Tc)
- aktualny desktop (< filename > . Dsk)
- aktualny projekt (< filename > . Prj)



1.3.10. Window (Okno).

Size/Move (Rozmiar / Przesuń)
Zoom (Powiększenie)
Tile (Obok siebie)
Cascade (Kaskadowo)
Next (Następne)
Close (Zamknij)
Close all (Zamknij wszystkie)
List all (Wykaz wszystkich aktywnych okien)

Okna, które można otwierać z menu Window:

Message	Wiadomości.
Output	Produkcji, wyjściowe.
Watch	Obserwacji.
User screen	Ekran użytkownika.
Register	Rejestry.
Project	Projekt.
Project notes	Notatki projektu.

1.3.11. Help (Pomoc).

Możemy przewinąć cierpliwie tekst opisu funkcji do końca. W opisach wielu funkcji na końcu umieszczony jest krótki „firmowy" program przykładowy. Nie musimy go przepisywać. W menu Edit IDE Borlanda mamy do dyspozycji polecenie:
Edit| Copy Example (skopiuj przykład) – przykład zostanie skopiowany do schowka.

1.4. Współpraca C ze środowiskiem operacyjnym na poziomie sprzętowym.

Współpraca ta nie jest objęta standardem ANSI C. Aby system C był systemem funkcjonalnie pełnym i w pełni otwartym, Borland wyposażył swój pakiet w następujące dodatkowe elementy, które są niezbędne do samodzielnego zbudowania systemu obsługi We/Wy dla kompilatora C:

- poza specyfikacją ANSI, C dodał słowo kluczowe asm pozwalające dołączyć kod pisany w asemblerze
- dodał dyrektywę #pragma inline pozwalającą na wstawianie do kodu w C wstawek asemblerowych. *(„Programowanie w języku C .(Miniprzewodnik).", A Majczak).*

Oto przykład jak posługiwać się danymi we wstawkach asemblerowych:

```
# pragma inline
void main()
{char *NAPIS="\n tekst - tekst $"; asm{
      MOV DX,NAPIS
      MOV AH, 9
      INT 33 }}
Lub{  _asm MOV DX,NAPIS
      _asm MOV AH, 9
      _asm INT 33 }
```

1.5. Pseudo zmienne.

Zdefiniował pseudo zmienne rejestrowe (register pseudo variables), pozwalające odwoływać się bezpośrednio do rejestrów procesora z poziomu kodu w C:

_AX unsigned int Akumulator
_AL unsigned char
_AH unsigned char
_BX unsigned int Rejestr indeksowy
_BL unsigned char
_BH unsigned char
_CX unsigned int Licznik cykli
_CL unsigned char
_CH unsigned char
_DX unsigned int Rejestr danych
_DL unsigned char
_DH unsigned char
_CS unsigned int Adr. Seg. kodu
_DS unsigned int Adr. Seg. danych
_SS unsigned int Adr. Seg. stosu
_ES unsigned int Adr. Seg. dodatkowy
_SP unsigned int Przemieszczenie względem SS
_BP unsigned int Przemieszczenie względem SS
_DI unsigned int Zmienna rejestrowa
_SI unsigned int Zmienna rejestrowa

Polecenie wykorzystujące te pseudo zmienne podczas wykonania przerwania to: geninterrupt(INT).

1.6. Modyfikator typu register, unia Regs.

Składnia: `register <definicja danych>;`
Mówi, aby kompilator przechowując zmienne będące rejestrami Cpu, optymalizował dostęp (jeżeli możliwy). Przykład: register int i;

`REGS (union) <DOS.H>`
Union Regs jest używana do przechowywania informacji do i od funkcji: Int86 int86x intdos intdosx

```
union REGS \{    struct  WORDREGS  x;
     struct  BYTEREGS  h;       \};
```

Przykład: `union REGS regs {regs.h.ah=...; regs.h.al.=...; regs.x.ax=...;}`

1.7. Przerwania – tablica przerwań.

Procesor posiada 256 różnych przerwań - można je podzielić na dwie grupy *(Rys. 1.2.)*:
Hardware interrupts - przerwania sprzętowe - przerwania wywoływane przez sprzęt jak np. klawiatura, zegar etc.

INT 08H Timer – Przerwanie zegara (IRQ 0) - wykonywane 18.2 razy na sekundę
INT 09H Keyboard – Przerwanie klawiatury (IRQ 1)
INT 0aH (reserved) – Zarezerwowane (IRQ 2)
INT 0bH Przerwanie łącza szeregowego (COM2 – IRQ 3)
INT 0cH Przerwanie łącza szeregowego (COM1 – IRQ 4)
INT 0dH Przerwanie drukarki (LPT2 – IRQ 5)
INT 0eH Diskette – Przerwanie sterownika dyskietki (IRQ 6)
INT 0fH Przerwanie drukarki (LPT1 – IRQ 7)

INT 70H zegar czasu rzeczywistego (IRQ 8)

Software interrupts - przerwania programowe - przerwania, które są wywoływane przez program - np. przerwanie 21h - przerwanie DOS-a. Również w przypadku przerwań programowych, po napotkaniu odwołania do przerwania (instrukcja INT) - procesor przerywa wykonywanie aktualnego programu i "przeskakuje" do procedury wywołanego przerwania.

Przykładowe przerwania to:

05H - bios - print screen - przerwanie uaktywniane, gdy naciśnięty zostaje klawisz

10H - bios obsługa karty graficznej

16H - bios - obsługa klawiatury

21H - dos - główne przerwanie funkcji/procedur

Każde z przerwań - bez względu na to, czy jest to przerwanie programowe czy sprzętowe - posiada procedurę obsługi - procedurę, która jest wykonywana w momencie wystąpienia danego przerwania - w przypadku przerwania klawiatury będzie to odczytanie znaku i jego zapis do bufora klawiatury. Procedury obsługi danego przerwania można oczywiście zmienić tak, żeby procesor po wystąpieniu przerwania wykonywał zamiast standardowej obsługi - np. część naszego programu. Po wykonaniu procedury obsługi danego przerwania procesor powraca do programu, który wykonywał przed wystąpieniem sygnału przerwania i wykonuje ten program.

Tablica przerwań.

W trybie rzeczywistym adresy procedur obsługi przerwań i wyjątków są zapisane w tablicy przerwań (ang. Interrupt vector table). Precyzyjnie tablica ta powinna być nazwana „tablicą wskaźników do procedur obsługi przerwań i wyjątków", gdyż rzeczywiście zawiera logiczne (selektor SEG i przemieszczenie OFS) adresy tych procedur. Jest to zbiór 256 dalekich wskaźników nazywanych wektorami, umieszczonych w pamięci poczynając od adresu 00:00h. Przerwania są ponumerowane kolejno – poczynając od 0. Numer przerwania (wyjątku) jest po prostu numerem elementu tablicy. *(„Encyklopedia informatyki", S Kruk).*

1.8. Notacja C – w stosunku do zapisu zmiennych.

Liczby całkowite dziesiętne mogą być użyte bezpośrednio, np.: x=2+107.

Liczby długie typu long powinny być inicjowane x=109L, po wykonaniu takiej instrukcji zmienna „x" zawiera liczbę dziesiętną 109.

Liczby całkowite szesnastkowe mogą być użyte bezpośrednio, np.: x=0x2+0x6B i tym razem po wykonaniu takiej instrukcji zmienna „x" zawiera liczbę dziesiętną 109 (0x6D). Przedrostek 0x jest obowiązkowy. Po tym właśnie przedrostku C rozpoznaje liczby szesnastkowe.

Znaki alfanumeryczne i ich kody ASCII: znak='B' lub znak=66 lub znak=0x42. Litera 'B' ma kod ASCII równy 66, więc za każdym razem zmienna będzie zawierać wartość 66 (dziesiętnie).

Łańcuchy Napis="ASD asd". Muszą występować ujęte w cudzysłów. W przeciwieństwie do Pascala, mogą liczyć więcej, niż 255 znaków. Kompilator C nie dodaje do łańcucha pola licznikowego (pierwszy bajt) określającego długość stringu. Kompilator C wykrywa koniec łańcucha znaków po specjalnym znaczniku końca łańcucha, oznaczonym '\0'.

1.9. Wewnętrzna reprezentacja danych w komputerze.

Komputer może analizować dane bit po bicie. Stany poszczególnych bitów są sprawdzane w postaci pól bitowych (bit fields), oraz w postaci flag. Nie chodzi tu tylko o stan poszczególnych pól rejestru FLAGS mikroprocesora.

Liczby całkowite są reprezentowane przez 8-bitowe, 16-bitowe lub 32-bitowe ciągi zer i jedynek. W zależności od lokalizacji tych danych powoduje to zajęcie:

- 8 bitów (1B)- połówka rejestru, np. AL, czy DH, pojedynczy bajt RAM o adresie [A],
- 16 bitów (2B)- cały rejestr np. AX, czy DX, dwa kolejne bajty pamięci o adresach [A] i [A+1]. W PC pod adresem [A] będzie młodszy bajt, pod adresem [A+1]-starszy bajt,
- 32 bajty (4B)-dwa rejestry 16b (np. AX i BX), jeden rejestr 32b (np. EAX), bajty RAM:[A], [A+1], [A+2], [A+3].

1.10. Opis podstawowych funkcji C do obsługi portów i przerwań.

(„Turbo C dla programistów.”, J Bielecki).

FP_OFF, FP_SEG, MK_FP < DOS . H >

Deklaracja :

```
unsigned FP_OFF(void far *p);
unsigned FP_SEG(void far *p);
void far *MK_FP(unsigned seg, unsigned ofs);
```

FP_OFF wyznaczenie przemieszczenia, wyodrębnienie tej części danej wskazującej reprezentowanej przez „p”, która stanowi przemieszczenie względem początku segmentu, zwraca wartość unsigned integer offset.
FP_SEG wyznaczenie numeru segmentu, zwraca wartość unsigned integer segment.
MK_FP utworzenie wskaźnika (pointer), którego numer segmentu ma wartość danej reprezentowanej przez seg, a przemieszczenie ma wartość ofs.

getvect , setvect < DOS . H >

Deklaracja :

```
void interrupt (*getvect(int interruptno))();
void setvect(int interruptno, void interrupt (*isr) ( ));
void interrupt(*_dos_getvect(unsigned interruptno)) ();
void _dos_setvect(unsigned interruptno, void interrupt (*isr) ());
```

getvect czyta wartość wektora przerwania dawanego przez Interruptno i wraca że wartość jest (dalekim) wskaźnikiem do przerwania
setvect umieszcza wartość wektor przerwania nazywanego przez Interruptno do nowej wartości , isr , który jest (dalekim) wskaźnikiem zawierającym adres nowej funkcji przerwania.
Interruptno - Wektor przerwania (wartości = 0 do 255).
getvect zwraca aktualną 4 bajtową wartość wprowadzaną do pamięci wektor przerwania nazywany przez interruptno, setvect nie zwraca .

inport , inportb , outport , outportb < DOS . H >

Deklaracja :

```
unsigned      inport  (unsigned portid);
unsigned char inportb (unsigned portid);
void          outport (unsigned portid, unsigned      value);
void          outportb(unsigned portid, unsigned char value);
```

inport – wprowadzenie danej słownej z portu o numerze portid
inportb – wprowadzenie danej bajtowej z portu o numerze portid
- inport i inportb zwraca wartość read
outport – wprowadzanie danej do portu słowowego o numerze portid danej reprezentowanej przez value.

outportb – wprowadzanie danej do portu bajtowego o numerze portid danej reprezentowanej przez value.
- outport i outportb nie zwraca wartości.

int86 , int86x < DOS . H >

Deklaracja :

```
int int86(int intno, union REGS *inregs, union REGS *outregs);
int int86x(int intno, union REGS *inregs, union REGS *outregs, struct
SREGS *segregs);
```

Int86 i int86x wykonuje przerwanie programowe 8086 wyszczególniane przez argument intno.
Int86 – Wygenerowanie przerwania programowego. Załadowanie rejestrów procesora danymi zawartymi w unii inregs, wygenerowania przerwania o numerze intno, skopiowanie danych zawartych w rejestrach procesora do unii outregs (m.in. skopiowanie bitu przeniesienia do pola x.cflag). W rezultacie dana typu int pozostawiona w rejestrze AX procesora.
Int86x – tak jak, powyżej, lecz przed tym tymczasowe zapamiętanie, a na końcu odtworzenie rejestru DS.

intdos , intdosx < DOS . H >

Deklaracja :

```
int intdos(union REGS *inregs, union REGS *outregs);
int intdosx(union REGS *inregs, union REGS *outregs, struct SREGS
*segregs);
```

Intdos – Wygenerowanie przerwania programowego DOS. Załadowanie rejestrów procesora danymi zawartymi w unii inregs, wygenerowanie przerwania o numerze 0x21 i skopiowanie danych zawartych w rejestrach procesora do unii outregs. W rezultacie dana typu int pozostawiona w rejestrze AX procesora.
Intdosx – tak jak, powyżej, lecz przed tym, tymczasowe zapamiętanie, a na końcu odtworzenie rejestru DS.

Disable, enable <DOS.H>

Deklaracje :

```
void disable(void);
void enable(void);
```

disable - uniemożliwia dostęp przerwaniom sprzętowym .
enable - pozwala przerywać zdarzenia, NMI (nie maskuje przerywania), jest dozwolone od jakiegokolwiek zewnętrznego urządzenia.

1.11. Funkcja do obsługiwania przerwań interrupt (przykład).

Definiuje funkcje jako przechwytującą, uchwyt przerwania (interrupt handler). W ogólnym przypadku, deklaracja zapowiadająca funkcję do obsługiwania przerwań powinna mieć postać *(Rys. 1.2.)*:

Składnia: **`interrupt <definicja funkcji>;`**

```
Void interrupt fun(bp, di, si, ds, es, dx, cx, bx, ax, ip, cs, flags...);
```

Jak wynika z tego zapisu, funkcja może mieć dostęp do wszystkich rejestrów procesora, identyfikowanych w obrębie jej definicji poprzez nazwy jej parametrów. Ponieważ funkcja może być wywoływana ze zmienną liczbą argumentów, nie jest wymagane ograniczenie się do argumentów przekazanych w rejestrach. Wszystkie rejestry CPU są ratowane i funkcja jest skończona z instrukcją Iret.
Oto przykład, gdzie cykliczne generowanie przerwania zegarowego 0x1C będzie wykonywać asynchronicznie funkcję timer:

```
#include<stdio.h>
#include<dos.h>
#define INT 0x1C
static volatile Count,Flag;
void interrupt far
timer()
{if(Count++%18) return; Flag++; }
main()
{void interrupt (far *Ref)();
unsigned i=0;
Ref=getvect(INT);
Setvect(INT,timer);
While(!kbhit())
{printf(„%8u”,i++); if(Flag) Flag--; putchar(‘\a’);}
setvect(INT,Ref);}
```