INFORMATYKA
KOMPUTERY
SYSTEMY

Dodatek „Żołnierza Wolności" Cena 200 zł

zeszyty programów komputerowych nr 4 /88

programy

JĘZYK 'C'

CZĘŚĆ 2

gry

DEEP BLUE C

Z. Dyjak
H. Krasuski

Kompilator DEEP BLUE C nie posiada zaimplementowanych liczb zmiennoprzecinkowych (dane typu float w standardowym C), co uniemożliwia, między innymi, wykorzystanie tego języka w poważniejszych obliczeniach matematyczno - technicznych czy w poważnych zastosowaniach graficznych, które są jedną z mocniejszych stron "małych ATARI". Niedostatek ten wypełniają biblioteki funkcji matematycznych i trygonometrycznych (MATHLIB i TRIG). Niniejszy zeszyt szczegółowo je przedstawia. Zamieszczone w końcowej części dwa programy: SPIRALE i OBLICZ ilustrują sposób posługiwania się funkcjami z tych bibliotek. Pokazują one również sposób łączenia procedur w kodzie maszynowym z programem w języku C oraz wykorzystanie jednego z ciekawszych mechanizmów tego języka - funkcji rekurencyjnych. Stanowią ponadto przykłady tworzenia stosunkowo dużych i skomplikowanych programów w "C".

SPIS TRESCI

01. WSTĘP

MATHLIB jest biblioteką tych funkcji języka C, które zapewniają realizację obliczeń zmiennoprzecinkowych z wykorzystaniem kompilatora DEEP BLUE C. Język DEEP BLUE C nie posiada zmiennoprzecinkowych typów danych. Uniemożliwia to, między innymi, wykorzystanie tego języka w poważnych zastosowaniach graficznych, które są mocną stroną komputerów Atari. Biblioteka MATHLIB wypełnia ten niedostatek DEEP BLUE C.

Wykorzystywanie funkcji biblioteki MATHLIB nie daje takiej zwięzłości jak używanie normalnych operatorów arytmetycznych, ale zapewnia pełny zakres możliwości obliczeniowych.

Program demonstracyjny, dołączony do biblioteki MATHLIB, ilustruje jak przy pomocy jej funkcji można zaimplementować grafikę żółwia.

Aby móc korzystać z funkcji biblioteki MATHLIB należy posiadać:

-kompilator DEEP BLUE C
-edytor ekranowy
-makroasembler Atari

Funkcje biblioteki MATHLIB działają na standardowych liczbach zmiennoprzecinkowych w formacie Atari i zapewniają dostęp do procedur zawartych w pamięci ROM. Dodatkowo dołączono funkcje trygonometryczne, stanowiące znaczne rozszerzenie możliwości procedur zawartych w ROM. W sumie MATHLIB zapewnia dodatkowe 32 funkcje.

02. SPIS FUNKCJI ZAWARTYCH W BIBLIOTECE MATHLIB

MATHLIB zapewnia realizację następujących funkcji:

- konwersja liczby podanej w kodach ATASCII na liczbę zmiennoprzecinkową i liczby zmiennoprzecinkowej na kody ATASCII;
- konwersja liczby całkowitej, bez lub ze znakiem, na liczbę zmiennoprzecinkową i liczby zmiennoprzecinkowej na liczbę całkowitą;
- dodawanie, odejmowanie, mnożenie i dzielenie liczb zmiennoprzecinkowych;
- pierwiastek kwadratowy;
- funkcje trygonometryczne (sinus, cosinus, tangens i arctangens) dla kątów podanych w stopniach i radianach;
- konwersja radianów na stopnie i stopni na radiany;
- zamianę stopni podanych w systemie dzesiętnym na stopnie, minuty, sekundy i na odwrót;
- wyodrębnianie części całkowitej i ułamkowej liczby zmiennoprzecinkowej;
- porównanie dwóch liczb zmiennoprzecinkowych;
- obliczanie modułu liczby zmiennoprzecinkowej;
- zmianę znaku liczby zmiennoprzecinkowej;
- nadanie liczbie zmiennoprzecinkowej wartości 0;
- przenoszenie wartości liczby zmiennoprzecinkowej z jednego miejsca pamięci w inne.

Zakładamy, że posiadasz kompilator DEEP BLUE C. Oto pliki DEEP BLUE C, które są wymagane podczas pracy z MATHLIB (nie są one dołączone do MATHLIB):

- CC.COM - kompilator DEEP BLUE C:
- CLINK.COM - konsolidator DEEP BLUE C;
- DBC.OBJ - moduł uruchomieniowy DEEP BLUE C:
- AIO.CCC - kod wynikowy funkcji we/wy.

Aby wygenerować wykonywalną postać programu demonstrującego grafikę żółwia (znajduje się na dyskietce zawierającej MATHLIB) należy dołączyć następujące pliki DEEP BLUE C (nie są dołączone do MATHLIB):

- GRAPHICS.CCC - kod wynikowy biblioteki funkcji graficznych;
- PRINTF.CCC - kod wynikowy funkcji wyprowadzania sformatowanego.

B3. PLIKI DOŁĄCZONE DO BIBLIOTEKI MATHLIB

Na dyskietce z biblioteką matematyczną znajdują się następujące pliki:

- MATHLIB.C - kod źródłowy funkcji matematycznych MATHLIB;
- MATHLIB.CCC - kod wynikowy dla MATHLIB.C, musi być zawsze wymieniony w pliku sterującym konsolidacją;
- TRIG.C - kod źródłowy funkcji trygonometrycznych;
- TRIG.CCC - kod wynikowy dla TRIG.C, musi być wymieniony w pliku sterującym konsolidacją, gdy korzystamy z funkcji trygonometrycznych;
- MATHLIB.OBJ - kod wynikowy dla napisanego w asemblerze interfejsu do procedur operacji zmiennoprzecinkowych zawartych w pamięci ROM. Musi być zawsze wymieniony w pliku sterującym konsolidacją. Plik MATHLIB.OBJ ładuje się do pamięci RAM w obszar od $2dc0 do $2fff. Jeżeli wykorzystujesz jednocześnie swoje własne procedury napisane w asemblerze, to upewnij się czy ich położenie nie koliduje z lokalizacją MATHLIB.OBJ. Możesz także, przy pomocy makroasemblera Atari, zmienić lokalizację MATHLIB.OBJ. W tym przypadku należy również zmienić adresy punktów wejściowych do MATHLIB.OBJ znajdujących się w MATHLIB.C i ponownie skompilować ten plik.
- MATHLIB.ASM - kod źródłowy w asemblerze dla MATHLIB.OBJ (wymaga użycia makroasemblera CX8121);
- TURTLE.C - kod źródłowy programu demonstracyjnego. Wykorzystuje funkcje matematyczne do realizacji podstawowych ruchów grafiki żółwia.
- TURTLE.CCC - kod wynikowy dla TURTLE.C;
- CLIPPER.C - kod źródłowy algorytmu Cohena-Southerlanda, adaptowanego dla języka DEEP BLUE C. Zawiera funkcje wywoływane przez TURTLE.
- CLIPPER.CCC - kod wynikowy dla CLIPPER.C;
- TURTLE.LNK - plik sterujący konsolidacją programu demonstracyjnego;
- TURTLE.COM - kod wykonywalny programu demonstrującego grafikę żółwia.

B4. WYKORZYSTANIE BIBLIOTEKI MATEMATYCZNEJ

W tym rozdziale przedstawiamy sposób wykorzystania biblioteki matematycznej. Pokażemy jak ją inicjować, jak deklarować zmienne zmiennoprzecinkowe, jak zamieniać liczby całkowite na zmiennoprzecinkowe i odwrotnie. Wszystkie nazwy funkcji zawartych w bibliotece zaczynają się przedrostkiem "c_" aby uniknąć konfliktów z nazwami użytkownika.

B4.1 Inicjalizacja biblioteki matematycznej.

Przed wykorzystaniem jakiejkolwiek funkcji z biblioteki matematycznej należy wywołać funkcję *c_iml*. Jeżeli wykorzystujemy funkcje trygonometryczne należy wywołać także *c_trig*. Funkcje te inicjują stałe i zmienne, których używają funkcje MATHLIB podczas realizacji obliczeń. Żadna z tych funkcji nie ma parametrów i są one wywoływane w sposób następujący:

```
c_iml();
c_trig();
```

Funkcja *c_trig* może być wywoływana tylko wówczas, gdy wykorzystywane są funkcje trygonometryczne, a plik TRIG.CCC został włączony do programu. Funkcja *c_iml* musi zostać wywołana przed *c_trig*.

B4.2 Deklaracja zmiennych zmiennoprzecinkowych.

Biblioteka MATHLIB wykorzystuje liczby zmiennoprzecinkowe przedstawione w standardowym formacie Atari. Aby móc wykorzystywać funkcje MATHLIB nie musisz znać formatu liczb zmiennoprzecinkowych w komputerze Atari i nie będziemy go tutaj omawiać. Jeżeli Cię to interesuje, to kierujemy do książki "*De Re Atari*", rozdział 8-45 i 8-46. Jedyne co należy wiedzieć to fakt, że liczby zmiennoprzecinkowe w Atari zajmują 6 bajtów. Funkcje biblioteki MATHLIB wykorzystują do przechowywania liczb zmiennoprzecinkowych tablice znakowe. Aby

zadeklarować zmienną, która będzie służyć do przechowywania liczby zmiennoprzecinkowej, napisz (zakładając, że nazwą tej zmiennej będzie *fpvar*):

```
char fpvar[6];
```

Wszystkie zmienne zmiennoprzecinkowe wykorzystywane prze funkcje biblioteki MATHLIB muszą być zadeklarowane w podany sposób, a wszystkie argumenty funkcji dotyczące liczb zmiennoprzecinkowych są wskaźnikami znakowymi. Oznacza to, że gdy w jakiejś funkcji z MATHLIB odwołujesz się do liczby zmiennoprzecinkowej używaj tylko nazwy zmiennej, bez indeksu. Na przykład jeżeli chcesz dodać dwie liczby zmiennoprzecinkowe *fa* i *fb* oraz chcesz aby wynik został zapamiętany w *fb*, to zmienne te powinny zostać zadeklarowane w sposób następujący:

```
char fa[6], fb[6];
```

Funkcja biblioteczna, która dodaje dwie liczby zmiennoprzecinkowe nazywa się *c_fadd()*. Wymaga podania trzech argumentów: dwie dodawane liczby i wynik.

Tak więc aby dodać dwie liczby w naszym przykładzie należy napisać:

```
c_fadd( fa, fb, fb);
```

Zwróć uwagę, że podajemy tylko nazwy zmiennych, bez indeksów.

B4.3 Tworzenie stałych zmiennoprzecinkowych.

DeeP BLUE C nie zapewnia standardowego typu danych *float*, a więc niemożliwe jest zadeklarowanie w tym języku stałej zmiennoprzecinkowej. Z tego też powodu stałe nie mogą być wprost przekazywane jako argumenty funkcji MATHLIB. Stała taka musi zostać najpierw stworzona przez odpowiednią funkcję biblioteczną, która faktycznie wpisuje ją do sześciobajtowej tablicy znakowej zadeklarowanej w programie użytkowym.

Stała zmiennoprzecinkowa jest początkowo łańcuchem znaków ATASCII, który następnie w jednym kroku zamieniany jest na standardową reprezentację zmiennoprzecinkową Atari. Zadanie to realizuje funkcja biblioteczna *c_afp()* (ATASCII to floating point). Funkcja *c_afp()* wykorzystuje jako argumenty dwa wskaźniki do tablic znakowych. Pierwszy wskazuje na łańcuch znaków ATASCII reprezentujący stałą, a drugi na sześciobajtową tablicę, która zawierać będzie zmiennoprzecinkową reprezentację tej stałej. Przypuśćmy, że chcemy stworzyć zmiennoprzecinkową reprezentację stałej "*pi*" (3.14159265) i zapamiętać ją w tablicy *fppi*. Należy napisać:

```
char fpi[6], *pntr;
pntr = "3.14159265";
c_afp( pntr, fppi);
```

W tym przykładzie *pntr* jest wskaźnikiem do łańcucha znaków ATASCII, a funkcja *c_afp()* przekształca go na reprezentację zmiennoprzecinkową, zapamiętaną następnie w sześciobajtowej tablicy znakowej *fppi*. *fppi* może być wykorzystywana w wywołaniach innych funkcji bibliotecznych i służyć do przekazywania wartości stałej "*pi*".

B4.4 Drukowanie liczby zmiennoprzecinkowej.

Wymiar tablicy dla reprezentacji liczby w kodzie ATASCII powinien być określony. Liczby zmiennoprzecinkowe w komputerze Atari zapewniają dziesięciocyfrową dokładność obliczeń. Dodatkowo liczba może być poprzedzona znakiem minus, zawierać kropkę, a także dwucyfrowy wykładnik w formie "E-xx". Na końcu liczby zmiennoprzecinkowej w kodzie ATASCII będzie występował znak *null* (standardowe zakończenie łańcucha w języku C). Dodając wszystko otrzymujemy 17 znaków i tak długa powinna być tablica znakowa służąca do przechowywania liczby zmiennoprzecinkowej w kodzie ATASCII.

Liczba zmiennoprzecinkowa w formacie Atari nie może być drukowana bezpośrednio. Musi zostać najpierw przekształcona na łańcuch znaków ATASCII. Tę czynność wykonuje funkcja biblioteczna *c_fasc()*. Posiada ona dwa argumenty będące wskaźnikami do łańcuchów znakowych. Pierwszy wskazuje na liczbę zmiennoprzecinkową a drugi na tablicę znakową, do której zostanie przekazana reprezentacja tej liczby w kodzie ATASCII. Tablica znakowa zawierająca przekształconą liczbę może być później wykorzystywana jako argument funkcji *printf()*.

Załóżmy na przykład, że mamy sześciobajtową liczbę w tablicy znakowej *fpnbr* i chcemy ją wydrukować wykorzystując funkcję *printf()*. Można tego dokonać w sposób następujący:

```
char fpnbr[6]; /* liczba zmiennoprzecinkowa */
char arfpnbr[17]; /* reprezentacja liczby w
 ATASCII */

c_fasc( fpnbr, arfpnbr); /* przekształcenie
liczby zmiennoprzecinkowej na kod ATASCII */
printf("%s", arfpnbr); /* wydruk reprezentacji
w  kodzie ATASCII */
```

B4.5.1 Konwersja liczby całkowitej na liczbę zmiennoprzecinkową.

Biblioteka MATHLIB zawiera także funkcje zapewniające konwersję liczb całkowitych na zmiennoprzecinkowe i na odwrót. Wyróżnia się dwa rodzaje liczb całkowitych: 16-bitowe bez znaku, z przedziału od 0 do 65535, i liczby ze znakiem

z przedziału od -32768 do +32767. Liczby bez znaku w DEEP BLUE C nie występują (liczba bez znaku większa niż 32767 traktowana jest jako ujemna). Jednakże jeżeli wiesz, że liczba zmiennoprzecinkowa nie jest ujemna to lepiej wykorzystywać funkcje biblioteki MATHLIB gdyż procedury ROM działają bezpośrednio na liczbach całkowitych bez znaku. Liczby całkowite ze znakiem wymagają dodatkowego przetwarzania.

Jeżeli wywołasz funkcję zamiany liczby zmiennoprzecinkowej bez znaku na liczbę całkowitą i liczba zmiennoprzecinkowa jest większa od 65535, to funkcja sygnalizuje błąd obliczeń. Taki sam błąd da próba zamiany liczby zmiennoprzecinkowej ze znakiem, większej niż 32767. Pamiętaj, że liczba całkowita bez znaku większa niż 32767 będzie traktowana przez DEEP BLUE C jak liczba ujemna.

Istnieją cztery procedury konwersji:

ufp() - konwersja liczby całkowitej bez znaku na liczbę zmiennoprzecinkową;

sfp() - konwersja liczby całkowitej ze znakiem na liczbę zmiennoprzecinkową;

c_fpi() - konwersja liczby zmiennoprzecinkowej bez znaku na liczbę całkowitą;

c_sfpi() - konwersja liczby zmiennoprzecinkowej ze znakiem na liczbę całkowitą.

B4.5.2 Forma opisu funkcji z biblioteki MATHLIB.

Kolejny rozdział zawiera opisy funkcji biblioteki MATHLIB. Każdy opis składa się z siedmiu części.

Zastosowanie.

Jedno lub dwa zdania precyzujące zastosowanie funkcji.

Wywołanie funkcji.

Przykład wywołania funkcji. Zdefiniowanie wszystkich parametrów jakie powinny zostać wprowadzone przed wywołaniem funkcji. Wyjątek stanowi *status* będący wartością zwracaną przez funkcję. Wartość zwracana określa czy dana operacja została wykonana poprawnie. Główną przyczyną błędu może być przekroczenie zakresu, tzn. wynikiem działania funkcji może być liczba przekraczająca standardowy zakres liczby zmiennoprzecinkowej w formacie Atari (od 10^{-99} do 10^{+98}). Jeżeli mamy pewność, że wynik będzie mieścił się w tym przedziale to nie musimy kontrolować wartości *status* po wywołaniu funkcji.

Parametry wejściowe.

Opisujemy tutaj wszystkie parametry przekazywane funkcji z programu wywołującego. Pierwszy wiersz każdego opisu zawiera nazwę parametru i jego typ. Potem następuje werbalny opis parametru.

Parametry wyjściowe.

Opisujemy tutaj wszystkie parametry przekazywane z funkcji do programu wywołującego.

Opis.

Przedstawiany jest tutaj opis czynności, jakie realizuje dana funkcja w odniesieniu do parametrów wejściowych i wyjściowych.

Wykorzystane funkcje.

W tym miejscu podajemy listę funkcji wykorzystanych do implementacji funkcji opisywanej. Mogą to być inne funkcje z biblioteki MATHLIB lub z biblioteki AIO. Jeżeli nie wykorzystywano innych funkcji oprócz standardowych konstrukcji języka, wystąpi w tym miejscu napis "Brak".

Przykład.

W tej części prezentujemy opcjonalnie jeden ze sposobów wykorzystania danej funkcji. Jeżeli opis funkcji jest bardzo czytelny tę część pomijamy. Przykłady zwykle rozpoczynają się od przedstawienia łańcuchów znaków ATASCII reprezentujących liczby zmiennoprzecinkowe, następnie obejmują przekształcenie ich na liczby zmiennoprzecinkowe w standardzie Atari, wywołanie funkcji, przekształcenie wyniku na kod ATASCII i jego wydruk przy pomocy funkcji *printf()*.

Jeżeli nazwa parametru używana jest w komentarzu, to pisana jest pismem pochyłym. Jeżeli parametr jest wskaźnikiem i brana jest pod uwagę wskazywana wielkość, to nazwę zmiennej poprzedza gwiazdka.

B4.6 Funkcje z biblioteki MATHLIB.

Funkcje opisane w tym rozdziale znajdują się w pliku MATHLIB.CCC. Wymieniamy je w kolejności ich występowania w dalszej części tego rozdziału.

c_iml inicjalizacja biblioteki MATHLIB.CCC;

c_afp konwersja z ATASCII na zmienny przecinek;

c_fasc konwersja ze zmiennego przecinka na ATASCII;

c_ifp konwersja liczby całkowitej bez znaku na zmiennoprzecinkową;

c_sifp konwersja liczby całkowitej ze znakiem na zmiennoprzecinkową;

c_fpi konwersja liczby zmiennoprzecinkowej na liczbę całkowitą bez znaku;

c_fpsi konwersja liczby zmiennoprzecinkowej

na liczbę całkowitą ze znakiem;

c_fadd dodawanie liczb zmiennoprzecinkowych;

c_fsub odejmowanie liczb zmiennoprzecinkowych;

c_fmul mnożenie liczb zmiennoprzecinkowych;

c_fdiv dzielenie liczb zmiennoprzecinkowych;

c_log logarytm naturalny liczby zmiennoprzecinkowej;

c_log10 logarytm dziesiętny liczby zmiennoprzecinkowej;

c_alog antylogarytm naturalny liczby zmiennoprzecinkowej;

c_alog10 antylogarytm dziesiętny liczby zmiennoprzecinkowej;

c_exp funkcja wykładnicza;

c_sqrt pierwiastek kwadratowy z liczby zmiennoprzecinkowej;

c_int część całkowita liczby zmiennoprzecinkowej;

c_frac część ułamkowa liczby zmiennoprzecinkowej;

c_cmp porównanie dwóch liczb zmiennoprzecinkowych;

c_abs moduł liczby zmiennoprzecinkowej;

c_chs zmiana znaku liczby zmiennoprzecinkowej;

c_zero ustawienie liczby zmiennoprzecinkowej na zero;

c_move przesunięcie liczby zmiennoprzecinkowej w pamięci;

B4.6.1 Inicjalizacja biblioteki.

Zastosowanie.

Do inicjalizacji stałych i zmiennych wymaganych przez bibliotekę do realizacji jej funkcji.

Wywołanie funkcji.

```
c_iml();
```

Parametry wejściowe.

Brak.

Parametry wyjściowe.

Brak.

Opis.

Funkcja ta musi zostać wywołana tylko raz przed wykorzystaniem jakiejkolwiek innej funkcji z biblioteki.

Wykorzystane funkcje.

c_afp;

B4.6.2 Konwersja kodu ATASCII na liczbę zmiennoprzecinkową.

Zastosowanie.

Przekształcenie liczby zmiennoprzecinkowej podanej w postaci łańcucha znaków ATASCII na format liczby zmiennoprzecinkowej w standardzie Atari.

Wywołanie funkcji.

```
status = c_afp( acs, fpn);
```

Parametry wejściowe.

acs char array

Wskaźnik do łańcucha znaków zawierającego reprezentację ATASCII liczby zmiennoprzecinkowej.

Parametry wyjściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na liczbę zmiennoprzecinkową w formacie Atari odpowiadającą liczbie wejściowej w podanej kodzie ATASCII.

status integer scalar

Zwracany status:

0 gdy liczba przekształcona poprawnie.

-1 gdy pierwszy bajt liczby w kodzie ATASCII jest niepoprawny.

Opis.

Funkcja ta pobiera kolejne bajty z **acs*, aż do napotkania bajtu, który nie może być częścią liczby. Bajty wprowadzone do tego momentu są przekształcane na liczbę zmiennoprzecinkową, która zostanie zapamiętana w tablicy **fpn* o długości sześciu bajtów. Jeżeli pierwszy bajt napotkany w **acs* jest niepoprawny to *status* będzie miał wartość -1. W przypadku przeciwnym przyjmuje wartość 0.

Wykorzystane funkcje.

Funkcja odwołuje się tylko do procedur zawartych w pamięci ROM.

Przykład.

```
char pntr, fpn[6];
pntr = "56.789";
c_afp( pntr, fpn);
```

B4.6.3 Konwersja liczby zmiennoprzecinkowej na liczbę w kodzie ATASCII.

Zastosowanie.

Przekształcenie liczby zmiennoprzecinkowej w standardzie Atari na łańcuch znaków w formacie odpowiednim do drukowania.

Wywołanie funkcji.

```
c_fasc( fpn, acs);
```

Parametry wejściowe.

fpn char arry

Wskaźnik do sześciobajtowej tablicy znaków,

która zawiera liczbę zmiennoprzecinkową w standardzie Atari.

Parametry wyjściowe.

acs char array

Wskaźnik do siedemnastobajtowej tablicy znaków, w której znajdzie się gotowa do wydruku reprezentacja liczby zmiennoprzecinkowej w kodzie ATASCII.

Opis.

Funkcja ta przekształca liczbę zmiennoprzecinkową *fpn* na postać gotową do wydruku (ATASCII) i umiesza ją w tablicy znakowej *acs*, która musi mieć przynajmniej 17 bajtów długości. Funkcja ta nie wykrywa żadnych błędów.

Wykorzystane funkcje.

Funkcja wykorzystuje bezpośrednio procedury zmiennoprzecinkowe zawarte w pamięci ROM.

Przykład.

```
char pntr, fpn[6], output[17];
pntr = "56.789";
c_afp( pntr, fpn);
c_fasc( fpn, output);
printf("%s", output);

56.789
```

B4.6.4 Zamiana liczby całkowitej bez znaku na liczbę zmiennoprzecinkową.

Zastosowanie.

Zamiana liczby całkowitej bez znaku (od 0 do 65535) na liczbę zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.

```
c_ifp( usint, fpn);
```

Parametry wejściowe.

usint integer scalar

Liczba całkowita bez znaku (od 0 do 65535), która ma zostać zamieniona na liczbę zmiennoprzecinkową.

Parametry wyjściowe.

fpn integer array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na liczbę zmiennoprzecinkową.

Opis.

Funkcja ta zamienia liczbę całkowitą bez znaku *usint* na liczbę zmiennoprzecinkową w formacie Atari i zapamiętuje ją w sześcibajtowej tablicy znakowej wskazywanej przez *fpn*. Funkcja nie wykrywa żadnych błędów. Zwróć uwagę, że DEEP BLUE C nie dopuszcza istnienia liczb całkowitych bez znaku. Wszystkie liczby całkowite większe niż 32767 są traktowane jak ujemne.

Wykorzystane funkcje.

Funkcja ta wykorzystuje bezpośrednio procedury zmiennoprzecinkowe zawarte w pamięci ROM.

Przykład.

```
int integer;
char fpn[6], output[17];
integer = -5000;
c_ifp( integer, fpn);
c_fasc( fpn, output);
printf("%s", output);

60536
```

B4.6.5 Zamiana liczby całkowitej ze znakiem na liczbę zmiennprzecinkową.

Zastosowanie.

Zamiana liczby całkowitej ze znakiem na liczbę zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.

```
c_sifp(sint, fpn);
```

Parametry wejściowe.

sint integer scalar

Liczba całkowita ze znakiem (od -32768 do +32767), która ma zostać zamieniona na liczbę zmiennoprzecinkową.

Parametry wyjściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której znajduje się wynikowa liczba zmiennoprzecinkowa.

Opis.

Funkcja ta przekształca liczbę całkowitą ze znakiem *sint* na liczbę zmiennoprzecinkową w formacie Atari i zapamiętuje ją w sześcibajtowej tablicy znakowej wskazywanej przez *fpn*. Funkcja nie wykrywa błędów.

Wykorzystane funkcje.

Funkcja wykorzystuje funkcję *c_ifp*, przez co jest od niej mniej efektywna w zastosowaniu do liczb dodatnich.

Przykład.

```
int integer;
char fpn[6], output[17];
integer = -5000;
c_sifp( integer, fpn);
```

```
c_ftasc( fpn, output);
printf("%s", output);

-5000
```

B4.6.6 Zamiana liczby zmiennoprzecinkowej na liczbę całkowitą bez znaku.

Zastosowanie.

Zamiana standardowej liczby zmiennoprzecinkowej podanej w formacie Atari na liczbę całkowitą bez znaku.

Wywołanie funkcji.

```
status = c_fpi( fpn, &usint);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej standardową liczbę zmiennoprzecinkową w formacie Atari, która ma zostać przekształcona na liczbę całkowitą bez znaku.

Parametry wyjściowe.

usint integr scalar

Zmienna całkowita przeznaczona na otrzymaną liczbę całkowitą bez znaku.

status integer scalar

Zwracany status:

0 - zamiana została wykonana poprawnie,

-1 - liczba zmiennoprzecinkowa jest >= 65535.5,

-2 - liczba zmiennoprzecinkowa jest ujemna.

Opis.

Funkcja ta zamienia standardową liczbę zmiennoprzecinkową w formacie Atari zawartą w *fpn* na liczbę całkowitą bez znaku. Jeżeli liczba zmiennoprzecinkowa jest ujemna zwracana jest wartość -2 i nie następuje żadna konwersja. Jeżeli liczba zmiennoprzecinkowa jest większa lub równa 65535.5 zwracana jest wartość -1 i, jak wyżej, konwersja się nie dokonuje. Funkcja ta realizuje prawdziwe zaokrąglanie, a nie obcinanie.

Wykorzystane funkcje.

Funkcja wykorzystuje bezpośrednio procedury działające na liczbach zmiennoprzecinkowych zawarte w pamięci ROM.

Przykład.

```
char pntr, fpn[6];
int integr;
pntr = "6000";
c_afp( pntr, fpn);
```

```
c_fpi(fpn, &integer);
printf("%d", integer);

-5536.
```

B4.6.7 Zamiana liczby zmiennoprzecinkowej na liczbę całkowitą ze znakiem.

Zastosowanie.

Zamiana standardowej liczby zmiennoprzecinkowej w formacie Atari na liczbę całkowitą ze znakiem.

Wywołanie funkcji.

```
status = c_fpsi( fpn, &sint);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową.

Parametry wyjściowe.

sint integer scalar

Zmienna całkowita przeznaczona na otrzymaną w wyniku konwersji liczbę całkowitą ze znakiem.

status integer scalar

Zwracany status:

0 - zamiana dokonała się poprawnie,

-1 - moduł liczby zmiennoprzecinkowej jest większy niż 32767.5.

Opis.

Funkcja zamienia liczbę zmiennoprzecinkową zawartą w *fpn* na liczbę całkowitą ze znakiem. Jeżeli moduł liczby jest większy lub równ. 32767.5, to zwracana jest wartość -1, a zamiana się nie dokonuje.

Wykorzystane funkcje.

Omawiana funkcja wywołuje *c_fpi*. W stosunku do liczb dodatnich jest ona mniej efektywna niż bezpośrednie wywołanie funkcji *c_fpi*.

Przykład.

```
char pntr, fpn[6];
int integr;
pntr = "60000";
c_afp( pntr, fpn);
c_fpsi(fpn, &integer);
printf("%d", integer);

-5536
```

B4.6.8 Dodawanie liczb zmiennoprzecinkowych.

Zastosowanie.

Dodanie dwóch liczb zmiennoprzecinkowych podanych w formacie [illegible].

Wywołanie funkcji.

```
status = c_fadd( fpn1, fpn2, fpsum);
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej pierwszą liczbę.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej drugą liczbę.

Parametry wyjściowe.

fpsum char array

Wskaźnik do sześciobajtowej tablicy, w której zostanie zapamiętana suma dwóch pierwszych liczb.

status integer scalar

Zwracany status:

0 - dodawanie wykonane poprawnie,

-1 - wynik przekroczył dopuszczalny zakres.

Opis.

Funkcja ta dodaje zawartość *#fpn1* do *#fpn2* i zapamiętuje wynik w *#fpsum*. Jeżeli wynik przekroczy zakres formatu liczb zmiennoprzecinkowych Atari, to zwracany jest status o wartości -1. Jeżeli operacja wykonana została poprawnie, zwracana jest wartość 0. *fp1*, *fp2* i *fpsum* mogą być tymi samymi wskaźnikami.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp(pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_add( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);
```

342.243

B4.6.9 Odejmowanie liczb zmiennoprzecinkowych.

Zastosowanie.

Odjęcie jednej liczby zmiennoprzecinkowej od drugiej.

Wywołanie funkcji.

```
status = c_fsub( odjemna, odjemnik, różnica);
```

Parametry wejściowe.

odjemna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej odjemną.

odjemnik char arry

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej odjemnik.

Parametry wyjściowe.

różnica char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której umieszczona zostanie obliczona różnica.

status integer scalar

Zwracany status:

0 - odejmowanie wykonane poprawnie,

-1 - wynik przekroczył dopuszczalny zakres.

Opis.

Funkcja ta odejmuje *#odjemnik* od *#odjemna* i zapamiętuje wynik w *różnica*. Jeżeli wynik przekracza dopuszczalny zakres liczb zmiennoprzecinkowych w formacie Atari, zwracana jest wartość -1. 0 oznacza prawidłowe wykonanie odejmowania.

Wykorzystane funkcje.

Funkcja ta wykorzystuje w sposób bezpośredni procedury znajdujące się w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp(pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c fsub(fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);
```

299.997

B4.6.10 Dzielenie liczb zmiennoprzecinkowych.

Zastosowanie.

Dzielenie jednej liczby zmiennoprzecinkowej przez drugą.

Wywołanie funkcji

```
status = c_fdiv( dzielna, dzielnik, iloraz);
```

Parametry wejściowe.

dzielna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej dzielną.

dzielnik char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej dzielnik.

Parametry wyjściowe.

iloraz char array

wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na wynik obliczeń.

status integer scalar

Zwracany status:

0 - poprawne zakończenie obliczeń.

1 - przekroczenie zakresu lub dzielenie przez zero.

Opis.

dzielna jest dzielona przez **dzielnik*, a wynik jest zapisywany w *iloraz* jeżeli wynik przekroczył dopuszczalny zakres lub wystąpiło dzielenie przez zero, to zwracana jest wartość -1. Jeżeli obliczenia są poprawne, zwracana jest wartość 0.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp( pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_div( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);
```

15.202386

B4.6.11 Mnożenie liczb zmiennoprzecinkowych.

Zastosowanie.

Pomnożenie przez siebie dwóch liczb zmiennoprzecinkowych.

Wywołanie funkcji.

```
status = c_fmul( mnożna, mnożnik, iloczyn);
```

Parametry wejściowe.

mnożna char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej mnożną.

mnożnik char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej mnożnik.

Parametry wyjściowe.

iloczyn char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której zostanie zapamiętany wynik mnożenia.

status integer scalar

Zwracany ststus:

0 - mnożenie wykonane poprawnie,

-1 - przekroczenie dopuszczalnego zakresu.

Opis.

Funkcja mnoży **mnożna* przez **mnożnik* i zapamiętuje wynik w *iloczyn*. Jeżeli nastąpi przekroczenie zakresu, zwrócona będzie wartość -1. 0 oznacza poprawne zakończenie obliczeń.

Wykorzystane funkcje.

Funkcja wykorzystuje w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, fp1[6], fp2[6], output[17];
pntr = "321.12";
c_afp( pntr, fp1);
pntr = "21.123";
c_afp( pntr, fp2);
c_fmul( fp1, fp2, fp2);
c_fasc( fp2, output);
printf("%s", output);
```

6783.01776

B4.6.12 Logarytm liczby zmiennoprzecinkowej.

Zastosowanie.

Wyliczanie logarytmu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
status = c_log( nbr, log);
status = c_log10( nbr, log);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której logarytm chcemy obliczyć.

Parametry wyjściowe.

log char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której zostanie zapamiętany logarytm liczby **nbr*.

status integer scalar

Zwracany status:

0 - obliczenia poprawne,

-1 - liczba ujemna lub przekroczenie zakresu.

Opis.

Funkcja *c_log()* daje w wyniku logarytm naturalny, a *c_log10()* logarytm zwykły. Jeżeli **nbr* jest ujemna lub powoduje przepełnienie, to status jest równy -1. Gdy obliczenia są poprawne, to funkcja zwraca wartość 0.

Wykprzystane funkcje.

Obydwie funkcje wykorzystują w sposób bezpośredni procedury zawarte w pamięci ROM.

Przykład.

```
char *pntr, nbr[6], log[6], answer[17];
pntr = "256.512";
c_afp( pntr, nbr);
c_log( nbr, log);
c_fasc( log, answer);
printf("%s", answer);

5.5471754
```

B4.6.13 Antylogarytm liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczanie antylogarytmu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
status = c_alog( nbr, antilog);
status = c_alog10( nbr, antilog);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową w formacie Atari, której antylogarytmu poszukujemy.

Parametry wyjściowe.

antilog char array

Wskaźnik do sześcibajtowej tablicy znakowej przeznaczonej na antylogarytm liczby **nbr*.

status integer scalar

Zwracany status:

0 - antylogarytm obliczony poprawnie,

-1 - przepełnienie.

Opis.

Funkcja *c_alog* oblicza antylogarytm naturalny, a *c_alog10* antylogarytm zwykły. Antylogarytm naturalny to e (2.7182818) podniesione do potęgi **nbr*. Antylogarytm zwykły to 10 podniesione do potęgi **nbr*. Jeżeli wystąpi przepełnienie, funkcja zwraca wartość -1. Jeżeli obliczenia zostały wykonane poprawnie, to funkcja zwraca wartość 0. *nbr* i *antilog* mogą być tym samym wskaźnikiem.

Wykorzystane funkcje.

Obydwie funkcje korzystają bezpośrednio z procedur zawartych w pamięci ROM.

Przykład.

```
char *pntr, nbr[6], log[6], answer[17];
pntr = "5.5471754";
c_afp( pntr, nbr);
c_alog( nbr, log);
c_fasc( log, answer);
printf( "%s", answer);

256.512
```

B4.6.14 Funkcja wykładnicza liczby zmiennoprzecinkowej.

Zastosowanie.

Podniesienie standardowej liczby zmiennoprzecinkowej do potęgi będącej również liczbą zmiennoprzecinkową w formacie Atari.

Wywołanie funkcji.

```
status = c_exp( podstawa, wykładnik, wynik);
```

Parametry wejściowe.

podstawa char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową w formacie Atari, której żądaną potęgę mamy znaleźć.

wykładnik char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową w formacie Atari, która ma być wykładnikiem potęgowania.

Parametry wyjściowe.

wynik char array

Wskaźnik do sześciobajtowej tablicy, w której zapamiętany zostanie wynik potęgowania liczby **podstawa*.

status integr scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zakresu.

Opis.

Liczba zawarta w *podstawa* zostanie podniesiona do potęgi *wykładnik* a wynik tej operacji umieszczony zostanie w *wynik*. *podstawa* i *wykładnik* mogą być tą samą zmienną, a *wynik* może być tą samą zmienną co *podstawa* lub *wykładnik*. Jeżeli wynik nie mieści się w zakresie liczb zmiennoprzecinkowych, funkcja zwraca wartość -1, a w przypadku przeciwnym wartość 0.

Wykorzystane funkcje.

c_fmul

c_log10

c_alog10

Przykład.

```
char *pntr, podstawa[6], wykladnik[6],
wynik[6], answer[17];
pntr = "2.37";
c_afp( pntr, podstawa);
pntr = "7.95";
c_afp( pntr, wykladnik);
c_exp( podstawa, wykladnik, wynik);
c_fasc( wynik, answer);
printf( "%s", answer);

953.34337
```

B4.6.15 Piewiastek kwadratowy z liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczyć pierwiastek kwadratowy ze standardowej liczby zmiennoprzecinkowej w formacie Atari.

Wywołanie funkcji.

```
status = c_sqrt( nbr, pierwiastek);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, której pierwiastek chcemy obliczyć.

Parametry wyjściowe.

pierwiastek char array

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostanie zapisany pierwiastek kwadratowy liczby *nbr.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,
-1 - przekroczenie zakresu,
-2 - liczba *nbr jest ujemna.

Opis.

Funkcja oblicza pierwiastek kwadratowy z dodatniej liczby zmiennoprzecinkowej znajdującej się w *nbr* i przesyła go do *pierwiastek*. Jeżeli pierwiastek obliczony został poprawnie to funkcja zwraca wartość 0, jeżeli liczba *nbr jest ujemna wartość -2, a jeżeli wynik obliczeń przekroczy dopuszczalny zakres wartość -1.

Wykorzystane funkcje.

c_fmul
c_log10
c_alog10

Przykład.

```
char *pntr, nbr[6], pierw[6], answer[17];
pntr = "256.512";
c_afp( pntr, nbr);
c_sqrt( nbr, pierw);
c_fasc( pierw, answer);
printf( "%s", answer);

16.015992001
```

B4.6.16 Część całkowita liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczenie części całkowitej liczby zmiennoprzecinkowej. Wynik jest także liczbą zmiennoprzecinkową.

Wywołanie funkcji.

```
status = c_int( nbr, intpor);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której część całkowitą chcemy uzyskać.

Parametry wyjściowe.

intpor char array

Wskaźnik do sześciobajtowej tablicy znakowej, w której znajdzie się część całkowita liczby *nbr.

status integer scalar

Zwracany status:

0 - liczba normalna,
-1 - brak części ułamkowej, do *intpor* przesłana jest *nbr*,
-2 - *nbr < 1, *intpor* jest ustawiana na zero.

Opis.

Część ułamkowa liczby *nbr* zostaje obcięta, a wynik zapamiętywany jest w *intpor*. Zwracana wartość niezerowa nie oznacza błędu tylko specjalne przypadki. *nbr* i *intpor* mogą być tą samą zmienną.

Wykorzystane funkcje.

move (biblioteka AIO)

Przykład.

```
char *pntr, nbr[6], intp[6], answer[17];
pntr = "1234.5678";
c_afp( pntr, nbr);
c_int( nbr, intp);
c_fasc( intp, answer);
printf( "%s", answer);

1234
```

B4.6.17 Część ułamkowa liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczenie części ułamkowej liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
status = c_frac( nbr, fracpor);
```

Parametry wejściowe.

nbr char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której część ułamkową należy obliczyć.

Parametry wyjściowe.

fracpor char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na część ułamkową liczby **nbr*.

status integer scalar

Zwracany status:

0 - liczba normalna,

-1 - **nbr* < 1, brak części całkowitej,

-2 - brak części ułamkowej, **fracpor* zostaje ustawiona na zero.

Opis.

Część całkowita liczby **nbr* zostaje obcięta, a wynik zapamiętywany jest w **fracpor*.

Niezerowa wartość statusu nie oznacza błędu, ale sygnalizuje specjalne przypadki wymienione wcześniej. **nbr* i **fracpor* mogą być tą samą zmienną.

Wykorzystane funkcje.

move (biblioteką AIO)

c_int

c_fsub

Przykład.

```
char *pntr, nbr[6], fracp[6], answer[17];
pntr = "1234.5678";
c_afp( pntr, nbr);
c_frac( nbr, fracp);
c_fasc( fracp, answer);
printf( "%s", answer);
```

```
0.5678
```

B4.6.18 Porównanie dwóch liczb zmiennoprzecinkowych.

Zastosowanie.

Porównanie dwóch liczb zmiennoprzecinkowych.

Wywołanie funkcji.

```
result = c_cmp( fpn1, fpn2)
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej pierwszą liczbę zmiennoprzecinkową.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej drugą liczbę zmiennoprzecinkową.

Parametry wyjściowe.

result char array

Wynik porównania:

-1 - **fpn1* mniejsza niż **fpn2*,

0 - **fpn1* równa **fpn2*,

+1 - **fpn1* większa niż **fpn2*.

Opis.

Liczba **fpn1* jest porównywana z **fpn2*. Jeżeli **fpn1* jest mniejsza niż **fpn2*, to *result* przyjmuje wartość -1. Jeżeli liczby są sobie równe, to result zostaje ustawiony na 0. Jeżeli **fpn1* jest większa niż **fpn2*, to *result* przyjmuje wartość +1.

Wykorzystywane funkcje.

c_fsub

Przykład.

```
char *pntr, nbr1[6], nbr2[6];
int status;


     pntr = "-27.45";
     c_afp( pntr, nbr1);
     pntr = "14.55";
     c_afp( pntr, nbr2);
     status = c_cmp( nbr1, nbr2);
     printf( "%s", status);

     -1
```

B4.6.19 Obliczanie modułu liczby zmiennoprzecinkowej.

Zastosowanie.

Obliczanie modułu liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
c_abs( fpn, absfpn);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której moduł chcemy obliczyć.

Parametry wyjściowe.

absfpn char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na moduł liczby **fpn*.

Opis.

Następuje obliczenie modułu liczby **fpn* i zapamiętanie go w **absfpn*. *fpn* i *absfpn* mogą być tą samą zmienną.

Wykorzystane funkcje.

Brak.

Przykład.

```
char *pntr, nbr[6], absnbr[6], answer[17];
pntr = "-15.789";
c_afp( pntr, nbr);
c_abs( nbr, absnbr);
c_fasc( absnbr, answer);
printf ("%s", answer);
```

15.789

84.6.20 Zmiana znaku liczby zmiennoprzecinkowej.

Zastosowanie.

Zmiana znaku liczby zmiennoprzecinkowej.

Wywołanie funkcji.

```
c_chs( fpn, negfpn);
```

Parametry wejściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, którą chcemy zanegować.

Parametry wyjściowe.

negfpn char array

Wskaźnik do sześcio bajtowej tablicy znakowej przeznaczonej na liczbę będącą negacją **fpn*.

Opis.

Zmieniony zostaje znak **fpn*, a wynik zostaje zapamiętany w **negfpn*. *fpn* i *negfpn* mogą być tą samą zmienną.

Wykorzystywane funkcje.

Brak.

Przykład.

```
char *pntr, nbr[6], output[6], answer[17];
pntr = "15.7895";
c_afp( pntr, nbr);
c_chs( nbr, output);
c_gasc( output, answer);
printf( "%s", answer);
```

-15.7895

84.6.21 Ustawienie liczby zmiennoprzecinkowej na zero.

Zastosowanie.

Ustawienie liczby zmiennoprzecinkowej na zero.

Wywołanie funkcji.

```
c_zero( fpn);
```

Parametry wejściowe.

Brak.

Parametry wyjściowe.

fpn char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową, której wartość wynosi zero.

Opis.

Do liczby **fpn* ładowane jest zero w formacie liczby zmiennoprzecinkowej Atari.

Przykład.

```
char fpn[6], answer[17];
c_zero( fpn);
c_fasc( fpn, answer);
printf ( "%s", answer);
```

0

84.6.22 Przesunięcie liczby zmiennoprzecinkowej.

Zastosowanie.

Przepisywanie liczb zmiennoprzecinkowych z jednego miejsca pamięci w inne.

Wywołanie funkcji.

```
c_move( fpn1, fpn2);
```

Parametry wejściowe.

fpn1 char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową, którą należy skopiować.

Parametry wyjściowe.

fpn2 char array

Wskaźnik do sześciobajtowej tablicy znakowej przeznaczonej na kopiowaną liczbę.

Opis.

Liczba **fpn1* jest kopiowana do **fpn2*.

Wykorzystywane funkcje.

move (biblioteka AIO)

Przykład.

```
char *pntr, fpn1[6], fpn2[6], answer[17];
```

```
pntr = "66";
c_afp( pntr, fpn1);
c_move( fpn1, fpn2);
c_fasc( fpn2, answer);
printf( "%s", answer);

66
```

B4.7 Funkcje trygonometryczne zawarte w pliku TRIG.CCC.

W tym rozdziale omówimy wszystkie funkcje trygonometryczne biblioteki MATHLIB zawarte w pliku TRIG.CCC. Oto ich lista:

c_trig	inicjalizacja funkcji trygonometrycznych;
c_rad	ustawienie trybu obliczeń;
c_rd	zamiana radianów na stopnie;
c_dr	zamiana stopni na radiany;
c_dmsd	zamiana stopni, minut, sekund na stopnie dziesiętne;
c_ddms	zamiana stopni dziesiętnych na stopnie, minuty, sekundy;
c_sin	obliczanie wartości funkcji sinus;
c_cos	obliczanie wartości funkcji cosinus;
c_tan	obliczanie wartości funkcji tangens;
c_atan	obliczanie wartości funkcji arctangens.

Określenie "stopnie dzesiętne" oznacza wartość podaną w stopniach, będącą liczbą zmiennoprzecinkową. Na przykład 30 stopni 25 minut i 37 sekund to 30.42694444 w stopniach dzesiętnych.

B4.7.1 Inicjalizacja funkcji trygonometrycznych.

Zastosowanie.

Inicjalizacja funkcji trygonometrycznych biblioteki MATHLIB.

Wywołanie funkcji.

```
c_trig();
```

Parametry wejściowe.

Brak.

Parametry wyjściowe.

Brak.

Opis.

Funkcja inicjalizuje stałe i zmienne używane przez funkcje trygonometryczne biblioteki MATHLIB. Musi zostać wywołana przed użyciem jakiejkolwiek funkcji trygonometrycznej. Pominięcie tej funkcji spowoduje, że obliczenia realizowane przez funkcje trygonometryczne będą błędne. Funkcja *c_trig* ustawia tryb wykonywania obliczeń w stopniach.

Wykorzystywane funkcje.

- c_afp

B4.7.2 Ustawienie trybu wykonywania obliczeń w stopniach lub radianach.

Zastosowanie.

Wybranie trybu realizacji obliczeń przez funkcje trygonometryczne. Obliczenia mogą być realizowane w stopniach lub radianach.

Wywołanie funkcji.

```
c_rad( flag);
```

Parametry wejściowe.

flag integer scalar

Parametr *flag* oznacza wybrany rodzaj pracy:

równy 0 - stopnie,

nie równy 0 - radiany.

Parametry wyjściowe.

Brak.

Opis.

Funkcja ta informuje MATHLIB, że obliczenia trygonometryczne będą realizowane w stopniach lub radianach. Funkcja ta może być wywoływana w dowolnym momencie. Wywołanie funkcji *c_trig* ustawia tryb pracy w radianach.

Wykorzystywane funkcje.

Brak.

B4.7.3 Zamiana radianów na stop nie.

Zastosowanie.

Zamiana radianów na stopnie.

Wywołanie funkcji.

```
status = c_rd( radiany, stopnie);
```

Parametry wejściowe.

radiany char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową określającą kąt w radianach.

Parametry wyjściowe.

stopnie char array

Wskaźnik do sześciobajtowej tablicy znakowej, do której przesłany zostanie kąt w stopniach dziesiętnych równy kątowi podanemu w *radiany*.

status integer scalar

Zwracany status:

0 - zamiana została dokonana poprawnie,

-1 - przekroczenie zakresu.

Opis.

Funkcja ta zamienia radiany na stopnie dziesiętne. *radiany* i *stopnie* mogą być tą samą zmienną.

Wykorzystywane funkcje.

c_fdiv

Przykład.

```
char *pntr, radiany[6], stopnie[6],
answer[17];
pntr = "0.78539816";
c_afp( pntr, radiany);
c_rd( radiany, stopnie);
c_fasc( stopnie, answer);
printf( "%s", answer);
```

45

B4.7.4 Zamiana stopni na radiany.

Zastosowanie.

Zamiana stopni dziesiętnych na radiany.

Wywołanie funkcji.

status = c_dr(stopnie, radiany);

Parametry wejściowe.

stopnie char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową oznaczającą kąt w stopniach dziesiętnych.

Parametry wyjściowe.

radiany char array

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostaje wpisany kąt w radianach.

status integer scalar

Zwracany status:

0 - zamiana wykonana poprawnie,

-1 - przekroczenie zakresu.

Opis.

Funkcja zamienia stpnie dziesiętne na radiany. *stopnie* i *radiany* mogą być tą samą zmienną.

Wykorzystywane funkcje.

c_fmul

Przykład.

```
char *pntr, radiany[6], stopnie[6],
answer[17];
pntr = "45";
c_afp( pntr, stopnie);
c_dr( stopnie, radiany);
c_fasc( radiany, answer);
printf( "%s", answer);
```

0.78539816

B4.7.5 Zamiana stopni, minut, sekund na stopnie dziesiętne.

Zastosowanie.

Zamiana stopni, minut i sekund na stopnie dziesiętne.

Wywołanie funkcji.

status = c_dmsd(stopnie, minuty, sekundy, sd);

Parametry wejściowe.

stopnie char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę stopni.

minuty char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę minut.

sekundy char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę sekund.

Parametry wyjściowe.

sd char array

Wskaźnik do sześciobajtowej tablicy znakowej, do której zostanie załadowana liczba określająca kąt w stopniach dziesiętnych.

status integer scalar

Zwracany status:

0 - zamiana dokonana poprawanie,

-1 - przekroczenie zakresu

Opis.

Kąt wyrażony w stopniach, minutach i sekundach zostaje zamieniony na stopnie dziesiętne.

Wykorzystane funkcje.

c_fdiv

c_fadd

Przykład.

```
char st[6], min[6], sek[6], std[6], *aux,
output[17];
aux = "30";
c_afp( aux, st);
aux = "25";
c_afp( aux, min);
aux = "37";
c_afp( aux, sek);
```

```
c_dmsd( st, min, sec, std);
c_fasc(ddeg, output);
printf("%s", output);

30.42694444
```

B4.7.6 Zamiana stopni dziesiętnych na stopnie, minuty, sekundy.

Zastosowanie.

Zamiana stopni dziesiętnych na stopnie, minuty i sekundy.

Wywołanie funkcji.

```
status = c_ddms( sd, stopnie, minuty, sekundy);
```

Parametry wejściowe.

sd char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową określającą kąt w stopniach dziesiętnych.

Parametry wyjściowe.

stopnie char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę stopni.

minuty char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę minut.

sekundy char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę sekund.

status integer scalar

Zwracany status:

0 - zamiana wykonana została poprawnie,

-1 - przekroczenie zakresu.

Opis.

Kąt podany w stopniach dziesiętnych zamieniany jest na kąt przedstawiony w stopniach, minutach i sekundach.

Wykorzystane funkcje.

c_int

c_fsub

c_fmul

Przykład.

```
char st[6], min[6], sek[6], std[6], *aux;
char out1[17], out2[17], out3[17];
aux = "30.42694444";
c_afp( aux, std);
c_DDms( std, st, min, sek);
c_fasc( st, out1);
c_fasc( min, out2);
c_fasc( sek, out3);
printf("%s %s %s", out1, out2, out3);

30 25 27
```

B4.7.7 Obliczanie wartości funkcji sinus.

Zastosowanie.

Obliczanie wartości funkcji sinus dla podanej wartości kąta.

Wywołanie funkcji.

```
status = c_sin( kąt, sinus);
```

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy zawierającej liczbę zmiennoprzecinkową określającą wartość kąta.

Parametry wyjściowe.

sinus char array

Wskaźnik do sześcobajtowej tablicy znakowej zawierającej liczbę zmiennoprzecinkową będącą wartością funkcji dla danego kąta.

status integer scalar

Zwracany status:

0 - obliczenia zostały wykonane poprawnie,

1 - przekroczenie zakresu.

Opis.

Kąt *kąt zostaje zredukowany do wartości z przedziału [0, +pi/2]. Wartość funkcji sinus, dzięki rozłożeniu jej w szereg Tayolra, obliczna jest z dokładnością do ośmiu miejsc po przecinku.

Wykorzystane funkcje.

move (biblioteka AIO)

c_fmul

c_fdiv

c_frac

c_fsub

c_fadd

Przykład.

```
char *pntr, nbr[6], sinnbr[6], answer[17];
rad(0); /* obliczenia w stopniach */
pntr = "30";
c_afp( pntr, nbr);
c_sin( nbr, sinnbr);
c_fasc( sinnbr, answer);
printf( "%s", answer);

0.5
```

B4.7.8 Obliczanie wartości funkcji cosinus.

Zastosowanie.

Obliczenie wartości funkcji cosinus dla zadanej wartości kąta.

Wywołanie funkcji.

status = c_cos(kąt, cosinus);

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej kąt podany w stopniach dzisiętnych.

Parametry wyjściowe.

cosinus char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość funkcji cosinus dla podanego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zkresu.

Opis.

Oblicza wartość funkcji cosinus dla kąta *k t i zapamiętuje ją w *cosinus*.

Wykorzystane funkcje.

move (biblioteka AIO)

c_fmul

c_fsub

c_sin

Przykład.

```
char *pntr, nbr[6], cosnbr[6], answer[17]
rad(0);  /* obliczenia w stopniach */
pntr = "30";
c_afp( pntr, nbr);
c_cos( nbr, cosnbr);
c_fasc( cosnbr, answer);
printf( "%s", answer);
```

0.8660254

B4.7.9 Obliczanie funkcji tangens.

Zastosowanie.

Obliczenie wartości funkcji tangens dla podanego kąta.

Wywołanie funkcji.

ststus = c_tan(kąt, tangens);

Parametry wejściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość kąta podaną w stopniach dziesiętnych.

Parametry wyjściowe.

tangens char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość funkcji dla zadanego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zakresu.

Wykorzystane funkcje.

c_sin

c_cos

c_fdiv

Przykład.

```
char *pntr, nbr[6], tannbr[6], answer[17];
rad(0);  /* obliczenia w stopniach */
c_afp( pntr, nbr);
c_tan( nbr, tannbr);
c_fasc( tannbr, answer);
printf( "%s", answer);
```

0.57735027

B4.7.10 Obliczanie wartości funkcji arctangens.

Zastosowanie.

Obliczanie wartości funkcji arctangens dla danej liczby rzeczywistej.

Wywołanie funkcji.

status = c_atan(tangens, kąt);

Parametry wejściowe.

tangens char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej liczbę, dla której należy policzyć wartość funkcji arctangens.

Parametry wyjściowe.

kąt char array

Wskaźnik do sześciobajtowej tablicy znakowej zawierającej wartość obliczonego kąta.

status integer scalar

Zwracany status:

0 - obliczenia wykonane poprawnie,

-1 - przekroczenie zakresu.

Opis.

Obliczenia realizowane są z dziewięciocyfrową dokładnością. Wynik mieści się w przedziale (-90, +90) w stopniach lub w przedziale (-pi/2, +pi/2) w radianach. Zależy to od wybranego trybu realizacji obliczeń.

Wykorzystane funkcje.

move (biblioteka AIO)

c_fdiv

c_cmp

c_fmul

c_fadd

```
        c_fsub

Przykład.
char *pntr, deg45[6], actan45[6], answer[17];
rad(0);  /* obliczenia w stopniach */
pntr = "1";
c_afp( pntr, deg45);
c_atan( deg45, atan45);
c_fasc( atan45, answer);
printf( "%s", answer);

45
```

PROGRAMY

*** SPIRALE ***

Program ten ilustruje wykorzystanie funkcji z biblioteki trygonometrycznej TRIG. Rozwiązuje on problem przedstawiony w książce Charlesa Corge p.t. ELEMENTY INFORMATYKI, PWN, Warszawa 1981r. Oddajmy głos autorowi tej pracy: "Cztery biedronki o imionach Afrodyta, Busiris, Cybela i Danaos znajdują się w danym momencie w czterech rogach kwadratu o boku a, oznaczonych odpowiednio ich inicjałami. Afrodyta i Cybela są samiczkami, Busiris i Danaos samcami. Każda z biedronek, pałając czułym, lecz niestety nie odwzajemnionym uczuciem, ożywiona jest nieodpartą chęcią spotkania się z wybrańcem swego serca. Decydują się na ten krok w tym samym momencie i wyruszają z identyczną prędkością; Afrodyta w kierunku Busirisa, Busiris w kierunku Cybeli, Cybela w kierunku Danaosa i wreszcie Danaos w kierunku Afrodyty". Problem polega na wykreśleniu dróg, które przebędą biedronki. Program SPIRALE kreśli te drogi. Z matematycznego punktu widzenia drogi te są krzywymi, które zakreślają wierzchołki kwadratu o boku a i środku znajdującym się w początku układu współrzędnych XY, poddanego serii przekształceń, będących złożeniem obrotu o kąt alfa z jednokładnością o stosunku exp(-alfa). Program obok funkcji sterującej main() składa się również z funkcji:

obl_jedn() - oblicza stosunek jednokładności, tj. wartość wyrażenia exp(-alfa) dla różnych wartości kąta alfa;

rek_kat() - dokonuje redukcji kąta z przedziału 0<=alfa<=360 stopni do kąta z przedziału 0-90 stopni;

obroc_kw() - kreśli kwadrat;

p_alfa() - wczytuje przyrost kąta obrotu alfa;

ut_tabs() - oblicza wartości funkcji sinus dla kątów od 0 do 90 stopni i umieszcza je w tabeli sinusów;

dr_ekran()- drukuje wykreślone drogi biedronek na drukarce ATARI 1029.

Ponadto program SPIRALE wykorzystuje procedurę w kodzie maszynowym (jest ona wywołana w funkcji dr_ekran()) drukującą graficzną zawartość ekranu. Przedstawiony kod źródłowy tej procedury należy zasemblować (na przykład asemblerem EASMD) i dołączyć kod maszynowy do programu podczas fazy linkowania.

Program SPIRALE

Funkcja sterujaca main()

```
/*************************************
 *                                   *
 *       program "SPIRALE"           *
 *                                   *
 *************************************

 * demonstruje wykorzystanie bibliotek
 * zmiennoprzecinkowej i
trygonometrycznej
 * kompilatora DBC w tworzeniu grafiki
 * w trybie wysokiej rozdzielczosci.
 * Program kresli figure powstala w
 * wyniku zlozenia przeksztalcen: obrotu
 * o kat alfa z jednokladnoscia o
stosunku
 * exp(-alfa) kwadratu o boku a i srodku
 * w poczatku ukladu wspolrzednych
kartezjanskich

 * Wazniejsze zmienne programu:
 * sint - tablica sinusow katow z
przedzialu
 *        0<=alfa<=90 stopni
 * kzred- tablica umozliwiajaca odczyt
 *        sinusa i cosinusa kata z w/w
 *        przedzialu
 * wspw - tablica wspolrzednych
wierzcholkow
 *        kwadratu
 * bok  - dlugosc boku kwadratu
 * alfa - kat obrotu kwadratu
 */


char sint[546];
```

```
main()
$(
 int wspw[8],kzred[4];
 char ars[20],c,c1;
 int
i,j,bok,czytaj,status,alfa,beta,xi,yi;
 char bokf[6],dwa[2],p2[6],pp2[6],sj[6];
 char rob[6],x[6],y[6],promien[6],e[6];
 char *const;
 int palfa,kont;
 kont=1;

/* inicjacja bibliotek */
 c_iml();
 c_itris();

/* utworz tabele sinusow */
 ut_tabs();

/* obliczenie wartosci stalej
0.5*sqrt(2) */
 const="2.0";
 c_afp(const,dwa);


 c_sqrt(dwa,p2);
 c_fdiv(p2,dwa,pp2);

 const="2.7183";  /* liczba e */
 c_afp(const,e);

 while(kont)
 $(

/* wczytaj dlugosc boku kwadratu */
 graphics(0);
 czytaj=1;
 while(czytaj)
  $(
   printf("\npodaj dlugosc boku
kwadratu\n");
   printf("40<=bok<=160\n");
   printf("bok = ");
   status=gets(arg);
   if (status<=0)
    $(
     printf("\g\nblad przy
wprowadzaniu\n");
     continue;
    $)
   bok=val(arg);
   if(bok<40 || bok>160)
    $(
     printf("\g\ndlugosc poza dozwolonym
zakresem");
     continue;
    $)
   czytaj=0;
  $)    /* koniec petli while  */

/* oblicz dlugosc polowy przekatnej
kwadratu */
/* tj. promien obrotu      */
 c_ifp(bok,bokf);
 c_fmul(bokf,pp2,promien);

/* wczytaj przyrost kata obrotu */
 palfa=p_alfa();

/* ustaw kolory  */
 graphics(24);
 color(1);
 poke(710,16);
 poke(712,16);
```

```
/* przeksztalcenie kwadratu i rysowanie
kwadratu
 * przeksztalconego  */
for(j=0;j<180;j+=palfa)
 $(
  obl_jedn(e,j,sj);
  for(i=0;i<=1;i++)
   $(
    beta=j+45+i*90;
    red_kat(beta,kzred);
    obroc_kw(kzred,promien,x,y);
    /* zmniejsz wspol. w skali sj */
    c_fmul(x,sj,rob);
    /* zamien wspol. na integer */
    c_fpsi(rob,&xi);
    wspw[2*i]=xi;

    c_fmul(y,sj,rob);
    c_fpsi(rob,&yi);
    wspw[2*i+1]=yi;
   $)
  wspw[4]=-wspw[0];
  wspw[5]=-wspw[1];
  wspw[6]=-wspw[2];
  wspw[7]=-wspw[3];
  rys_kw(wspw);
 $)
while(peek(53279)!=5);
dr_ekran();
graphics(0);
printf("chcesz jeszcze raz (T/N)");
c=toupper(getchar());
c1=getchar();
if(c=='N') kont=0;
$)
```

Funkcja obl_jedn()

```
/*   FUNKCJA obl_jedn(e,alfa,stjedn)
 * oblicza wartosc wyrazenia
 *    stjedn=exp(-alfa)
 * ktora jest wykorzystana jako skala
 * zmniejszenia dlugosci boku kwadratu
 * przy jego obrocie o kat alfa
 */

obl_jedn(e,alfa,stjedn)
int alfa;
char *stjedn,*e;
$(
 char alfaf[6],alfar[6];
 c_ifp(alfa,alfaf);     /* zamien int na
fp */
 c_dr(alfaf,alfar);     /* zamien stopnie
na radiany */
 *alfar = *alfar^0x80; /* zmien znak   */
 c_exp(e,alfar,stjedn);
$)   /* koniec funkcji obl_jedn() */
```

Funkcja red_kat()

```
/*      Funkcja red_kat(kat,kzred)
 * dokonuje redukcji kata z przedzialu
 * 0<=kat<=360 stopni do kata z
przedzialu
 * 0<=kat<=90 stopni.
 * miara zredukowanego kata nie ulega
zmianie
 * funkcja umieszcza cztery wielkosci
typu
```

```
 * integer w czteroelementowej tablicy
KZRED
 * ich znaczenie jest nastepujace:
 * kzred[0] - liczba wskazujaca element
tablicy sint
 * zawierajacy sinus kata
 * kzred[1] - j.w. dotyczy cosinusa kata
 * kzred[2] - znak sinusa (kzred[2]=1
gdy sin<0,
 *                          kzred[2]=0

gdy sin>=0)
 * kzred[3] - znak cosinusa (podobnie
jak wyzej)
 */


red_kat(kat,kzred)
int kat;
int *kzred;
${
 int m,n,znaks,znakc;
 if (kat>359) kat=kat%360;
 if (kat<91)
  ${
   m=kat;
   znaks=0;
   n=90-kat;
   znakc=0;
  $}
 else if (kat<181)
  ${
   m=180-kat;
   znaks=0;
   n=kat-90;
   znakc=1;
  $}
 else if (kat<271)
  ${
   m=kat-180;
   znaks=1;
   n=270-kat;
   znakc=1;
  $}
 else
  ${
   m=360-kat;
   znaks=1;
   n=kat-270;
   znakc=0;
  $}
 *kzred=m;
 *(kzred+1)=n;
 *(kzred+2)=znaks;
 *(kzred+3)=znakc;
$}  /* koniec funkcji red_kat()  */
```

Funkcja obroc_kw()

```
/* FUNKCJA obroc_kw(kzred,promien,x,y)
 * wyznacza wspolrzedne jednego
wierzcholka
 * kwadratu po obrocie o kat, ktorego
 * sinus i cosinus (i ich znaki)
okreslane sa
 * przez elementy tablicy KZRED
 * promien - zawiera dlugosc polowy
 * przekatnej kwadratu w postaci liczby
 * zmiennoprzecinkowej.
 * x,y - wspolrzedne po obrocie
 */


obroc_kw(kzred,promien,x,y)
int *kzred;
char *promien,*x,*y;
${


char *wsk;
int j;
char sin[6],cos[6];
j = *kzred;
wsk=sint + j*6;
move(wsk,sin,6);
j = *(kzred+2);
if(j) *sin = *sin^0x80;
j = *(kzred+1);
wsk=sint + j*6;
move(wsk,cos,6);
j = *(kzred+3);
if(j) *cos = *cos^0x80;
c_fmul(cos,promien,x);
c_fmul(sin,promien,y);
$}  /* koniec funkcji obroc_kw()  */
```

Funkcja p_alfa()

```
/*        FUNKCJA P_ALFA()
 * wczytuje przyrost kata obrotu alfa
 */

p_alfa()
${
 int znak;
 int iocb,pob;
 printf("\n\nWybierz przyrost kata
obrotu alfa\n");
 printf("\262  - dwa stopnie\n");
 printf("\263  - trzy stopnie\n");
 printf("\264  - cztery stopnie\n");
 printf("\265  - piec stopni\n");
 printf("\266  - szesc stopni\n");
 printf("\267  - siedem stopni\n");
 position(2,18);
 printf("po narysowaniu figury nacisnij
\323\305\314\305\303\324");
 iocb=copen("K:",'r');
 pob=1;
 while(pob)
  ${
   znak=cgetc(iocb);
   if(znak<'2' || znak>'7')
    ${
     printf("\a");
     continue;
    $}
   pob=0;
  $}
 cclose(iocb);
 return(znak-48);
$}
```

Funkcja dr_ekran()

```
/*     funkcja dr_ekran()
 * drukuje zawartosc pamieci ekranu
 * (w 8 trybie graficznym) na drukarce
 * funkcja wykorzystuje podprogram
 * w kodzie maszynowym rezydujacy na
 * szostej stronie pamieci (adres 1536
 * lub 0x600)
 */

 dr_ekran()
 ${
```

```
int l,iocb;
char *ekran;
ekran=dpeek(88);
for(l=0;l<28;l++)
 $(
/*
 * otwarcie kanalu dla drukarki
 */
  iocb=copen("P:",'w');
  if(iocb<0)
   $(
    graphics(0);
    printf("\g\n\ndrukarka odlaczona");
    return;
   $)
/* ustaw tryb graficzny pracy drukarki
*/
  cputc('\033',iocb);
  cputc('9',iocb);
  cputc('\033',iocb);
  cputc('A',iocb);
  cputc('\001',iocb);
  cputc('\030',iocb);
/*
 * wywolanie podprogramu rezydujacego
 * na szostej stronie pamieci
 */

  usr(1536,iocb,ekran);

  cclose(iocb);
  ekran=ekran+280;
 $)
$)      /* koniec funkcji dr_ekran() */
```

Funkcja ut_tabs()

```
/*
 * FUNKCJA ut_tabs()
 * oblicza wartosci funkcji sinus
 * dla katow od 0 do 90 stopni i
 * umieszcza je w postaci szescio-
 * bajtowych liczb zmiennoprzecinkowych
 * w tablicy sint[594]
 */

ut_tabs()
$(
 int i;
 char kat[6],sinasc[17];
 char *wsksin;
 printf("\f");   /* czysc ekran */
 poke(752,1);    /* schowaj kursor */
 position(10,3);
 printf("**********************");
 position(10,4);
 printf("*                    *");
 position(10,5);
 printf("*  program SPIRALE   *");
 position(10,6);
 printf("*                    *");
 position(10,7);
 printf("**********************\n\n");
 printf(" kresli cztery spirale
logarytmiczne\n");
 printf("   stanowiace rozwiazanie
problemu\n");
 printf(" przedstawionego w pracy C.
Corge\n");
 printf("      p.t. Elementy
Informatyki\n");
 printf("    PWN, Warszawa 1981r. str.
340\n\n");
 printf("Prosze chwile
poczekac...\n\n");
 printf("tworzenie tabeli sinusow");
 i=0;
 c_rad(0);     /* miary katow - stopnie
*/
 while(i<91)
  $(
   c_ifp(i,kat);     /* zamien integer na
fp */
   wsksin=sint+6*i; /* wskaznik do
odpowiedniego */
                    /* elementu tablicy
sint     */
   c_sin(kat,wsksin);
   c_fasc(wsksin,sinasc);
   position(1,20);
   printf("kat %-3d stop.   sin =
%s",i,sinasc);
   ++i;
  $)       /* koniec while */
$)         /* koniec funkcji ut_tabs() */
```

```
01 ;PR1029.ASM
05 .OPT OBJ
10 ;Podprogram wyprowadza na drukarke
20 ;ATARI 1029 zawartosc fragmentu
30 ;pamieci ekranu w 8 trybie graficznym
40 ;Sposob wywolania z jezyka DBC
50 ;
60 ; usr(adr,iocb,adres)
70 ; gdzie adr,iocb sa zmiennymi
80 ; typu int i oznaczaja:
90 ; adr   - poczatek obszaru pamieci
RAM,

0100 ;          w ktorym rezyduje ten
podprogram
0110 ;iocb - numer IOCB, z ktorym
zostala
0120 ;          otworzona drukarka.
Podprogram nie
0130 ;          przeprowadza operacji
otwarcia.
0140 ;adres- adres pamieci ekranu,
ktorego
0150 ;          zawartosc ma byc
wydrukowana.

0152 ;          "adres jest wskaznikiem
typu char.
0160 ;UWAGA: podprogram wyprowadza
jedynie
0170 ;          280 bajtow pamieci ekranu
poczawszy
0180 ;          od adresu "adres"
0190 ;
0200 ;
0210 ;Wykorzystane zmienne systemu
operacyjnego

0220 ICBCOM = $0342
0230 ICBAL  = $0344
0240 ICBAH  = $0345
0250 ICBLL  = $0348
0260 ICBLH  = $0349
0270 CIOV   = $E456
0280 ;Zmienne podprogramu
```

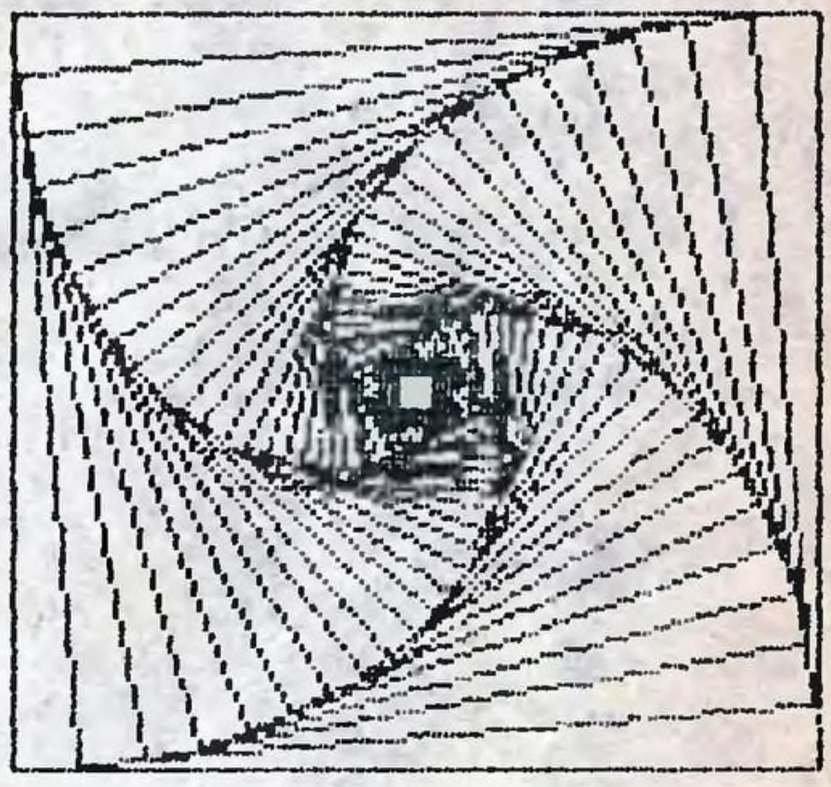
```
0290 IOCB   = $06E0
0300 ADRES  = $06E2
0305 EKRAN  = $FFFF
0310 OBSZ   = $06E4
0320 ;
0330 ;Poczatek podprogramu
0340 ;
0350        *= $0600
0360 ;
0370 START  PLA     ;zwieksz wskaznik
stosu
0380        PLA     ;st. bajt nr IOCB
0390        PLA     ;ml. bajt nr IOCB
0400        ASL A   ;pomnoz
0410        ASL A   ;przez
0420        ASL A   ;szesnascie
0430        ASL A   ;
0440        STA IOCB
0450        LDA #0      ;zeruj starszy
bajt
0460        STA IOCB+1  ;numeru IOCB
0470        PLA     ;pobierz ADRES
0480        STA ADRES+1
```



```
0490        PLA
0500        STA ADRES
0510 DALEJ  JSR AKTAD ;aktualizuj adresy
0520        CLC       ;i drukuj po 8
bajtow
0530        LDA PRZES+1
0540        ADC #1
0550        STA ADRES
0560        LDA PRZES+2
0570        ADC #0
0580        STA ADRES+1
0590        INC IOCB+1
0600        LDA IOCB+1
0610        CMP #40
0620        BNE DALEJ
0630 KONIEC RTS  ;wydrukowano 280 bajtow
0640 AKTAD  LDX #0
0650 AKTAD1 CLC
0660        LDA ADRES+1
0670        STA PRZES+2,X
0680        LDA ADRES
0690        STA PRZES+1,X
0700        ADC #40
0710        STA ADRES
0720        LDA ADRES+1
0730        ADC #0
0740        STA ADRES+1
0750        CLC
0760        TXA
0770        ADC #6
0780        TAX
0790        CPX #37
0800        BCC AKTAD1
0810        LDX #0
0820 PRZES  ASL EKRAN
0830        ROL OBSZ,X
0840        ASL EKRAN
0850        ROL OBSZ,X
0860        ASL EKRAN
0870        ROL OBSZ,X
0880        ASL EKRAN
0890        ROL OBSZ,X
0900        ASL EKRAN
0910        ROL OBSZ,X
0920        ASL EKRAN
0930        ROL OBSZ,X
0940        ASL EKRAN
0950        ROL OBSZ,X
0960        INX
0970        CPX #8
0980        BNE PRZES
0982 ;
0984 ;druku.j osiem bajtow
0986 ;
0990 DRUK   LDX IOCB
1000        LDA #11
1010        STA ICBCOM,X
1020        LDA #OBSZ&255
1030        STA ICBAL,X
1040        LDA #OBSZ/256
1050        STA ICBAH,X
1060        LDA #8
1070        STA ICBLL,X
1080        LDA #0
1090        STA ICBLH,X
1100        JSR CIOV
```



```
1110        RTS
1120        .END
```



*** OBLICZ ***

Ostatni z prezentowanych programów ilustruje sposób tworzenia i wykorzystania w języku C funkcji rekurencyjnych. Funkcja rekurencyjna jest to funkcja, która rozwiązując problem, sama siebie wywołuje. Program OBLICZ oblicza wartość złożonego wyrażenia matematycznego (ciąg liczb połączonych znakami operacji matematycznych np: 2.0+2.0-1+1000).

Dozwolonymi znakami operacji matematycznych są: +. -.*,:. Funkcją rekurencyjną jest funkcja obl(). Analizuje ona napisane na ekranie wyrażenie i po napotkaniu znaku operacji matematycznej wywołuje sama siebie w celu zanalizowania części wyrażenia, stojącego po prawej stronie operatora. Funkcja obl() wykonuje najpierw mnożenie, następnie dzielenie, następnie dodawanie, a na końcu odejmowania, stąd wartość wymienionego wyżej przykładowego wyrażenia obliczona przez funkcję obl() jest równa -997. Używając nawiasów (), można zmienić powyższą hierarchię działań.

Funkcje rekurencyjne są bardzo wygodnym narzędziem oferowanym przez język C. Dzięki nim można napisać krótką i zwartą funkcję, rozwiązującą jednak stosunkowo skomplikowany problem (proszę spróbowac napisać podobną do funkcji obl() procedurę w języku BASIC i porównać je). Pisząc funkcję rekurencyjną, należy zawsze szacować liczbę jej wywołań, konieczną do rozwiązania problemu, dla którego funkcja ta jest tworzona. Zbyt duża liczba wywołań tzw. "nieskończona rekurencja", może doprowadzić do "przepełnienia" stosu i zawieszenia się programu wynikowego.

Program OBLICZ

Program "OBLICZ"

Funkcja sterujaca main()

```
/**********************************
 *       PROGRAM "OBLICZ"         *
 * demonstruje dzialanie funkcji  *
 * rekurencyjnej w jezyku C       *
 **********************************/

main()
$(
char
wiersz[39],kom1[35],kom2[35],kom3[35];
char kom4[35];
char wynik[17],wynikf[6];
char *vram,*kom;
char c,c1;
int stat,dalej;
c_iml();
kom="Nieparzysta liczba nawiasow ()";
strcpy(kom1,kom);
kom="   Blad w podanym wyrazeniu.   ";
strcpy(kom2,kom);
kom="Dlugosc przekroczyla 38 znakow";
strcpy(kom3,kom);
kom="   Chcesz liczyc dalej T/N     ";
strcpy(kom4,kom);
poke(752,1);
printf("\f\g");    /* czysc ekran */
pisz(5,1,"Podaj dowolne wyrazenie",0);
pisz(1,2,"zbudowane z liczb, nawiasow ()
oraz",0);
pisz(1,3,"symboli operacji
matematycznych tj.",0);
pisz(15,4,"+, -, *, :",0);
pisz(1,5,"mozesz uzywac liczb z kropka
dzies.",0);
pisz(15,7,"WYRAZENIE",128);
ryskon(0,9,39,11,'c');
pisz(15,13,"WARTOSC",128);
ryskon(10,15,31,17,'g');
vram=dpeek(88)+401;
/*
 * wczytaj wyrazenie
 */
dalej=1;
while(dalej)
 $(
  stat=wczytaj_w(vram,wiersz);
  if(stat)
   $(
    pisz(2,20,kom3,0);
    printf("\g");
    continue;
   $)
  stat=spr_naw(wiersz);
  if(stat)
   $(
    pisz(2,20,kom1,0);
    printf("\g");
    continue;
   $)
 /*
  * oblicz wartosc wyrazenia


 */
  stat=obl(wiersz,wiersz+strlen(wiersz)-
1,wynikf);
  if(stat)
   $(
    pisz(2,20,kom2,0);
    printf("\g");
    continue;
   $)
 /*
  * wyprowadz wartosc wyrazenia
  */
  c_fasc(wynikf,wynik);
  position(12,16);
  printf("                    ");
  position(12,16);
  printf("%s",wynik);
  pisz(2,20,kom4,0);
  c=toupper(getchar());
  c1=getchar();
  if(c!='T') dalej=0;
 $)  /* koniec petli while */
$)   /* koniec funkcji main() */
```

Funkcja spr_znak()

```
#define DEL 126
#define RET 155

/**********************************
 *     funkcja spr_znak(znak)     *
 * sprawdza, czy wczytany znak jest*
 * jednym ze znakow DEL, RETURN, -,*
 * +,*,:,.,(,) lub cyfra. Jezeli  *
 * tak funkcja zwraca zero do     *
 * miejsca wywolania, w przeciwnym *
 * wypadku zwraca -1              *
 **********************************/

spr_znak(znak)
char znak;
$(
 if(znak==DEL) return(0);
```

```
 if(znak==RET) return(0);
 if(znak=='-') return(0);
 if(znak=='+') return(0);
 if(znak=='*') return(0);
 if(znak==':') return(0);
 if(znak=='.') return(0);
 if(znak=='(') return(0);
 if(znak==')') return(0);
 if(znak>='0' && znak<='9') return(0);
 return(-1);
$)
```

Funkcja wczytaj_w()

```
/*************************************
 *  funkcja wczytaj_w(adr,wier)      *
 * wczytuje wiersz (wyrazenie),      *
 * wartosc ma byc obliczona. Zwraca  *
 * 0 gdy wiersz zostal wprowadzony   *
 * poprawnie, -1 gdy jego dlugosc    *
 * przekroczyla 38 znakow            *
 *************************************/

wczytaj_w(adr,wier)
char *adr,*wier;
$(
 char znak;
 int czyt,klaw,rz,lpz,i;
/*
 * umieszczenie 38 znakow spacji w
inwersie
 * w pamieci obrazu od adresu adr
 */
 for(i=0;i<38;i++)
     poke(adr+i,128);
 klaw=copen("K:",'r');
 czyt=1;
 lpz=0;
 while(czyt)
  $(
if(lpz>37) return(-1);
znak=cgetc(klaw);
rz=spr_znak(znak);
if(rz)   /* gdy zly znak */
 $(
  printf("\a");
  continue;
 $)
if(znak==DEL && lpz>0)
 $(
  adr--;
  wier--;
  lpz--;
  poke(adr,128);
  continue;
 $)
if(znak==DEL && lpz==0)
 $(
  printf("\a");
  continue;
 $)
if(znak==RET)
 $(
  *wier='\0';
  cclose(klaw);
  return(0);
 $)
poke(adr,(znak-32)|128);
*wier=znak;
adr++;
wier++;
lpz++;
  $)     /* koniec petli while */
$)       /* koniec funkcji wczytaj_w() */
```

Funkcja spr_naw()

```
/*************************************
 *      funkcja spr_naw(wiersz)      *
 * sprawdza, czy w podanym wyrazeniu*
 * wystepuje parzysta liczba nawia- *
 * sow zamykajacych ')' i otwieraja-*
 * cych '('. Zwraca 0 gdy jest tak  *
 * a -1 w przeciwnym wypadku        *
 *************************************/

spr_naw(wiersz)
char *wiersz;
$(
int lnaw;
lnaw=0;
while(*wiersz)
  $(
   if(*wiersz=='(') lnaw++;
   if(*wiersz==')') lnaw--;
   wiersz++;
  $)
if(lnaw) return(-1); else return(0);
$)  /* koniec funkcji spr_naw() */
```

Funkcja obl()

```
/*************************************
 *  funkcja obl(pocz,kon,liczba)     *
 * funkcja rekurencyjna obliczjaca   *
 * wartosc wyrazenia.                *
 * argumenty:                        *
 * pocz - wskaznik do poczatku       *
 *        tablicy znakowej zawiera-  *
 *        jecej wyrazenie            *
 * kon  - wskaznik do konca wyraze-  *
 *        nia; kon=pocz+dlugosc-1    *
 * liczba-wskaznik do tablicy prze-  *
 *        znaczonej na wartosc wyr.  *
 *************************************/

obl(pocz,kon,liczba)
/* zmienne zewnetrzne */

char *pocz,*kon,*liczba;

$(
char liczba1[6],liczba2[6];
char *i;
int poziom;

poziom=0;
if(pocz>kon+1) return -1;
if(pocz==kon+1)$(
  c_zero(liczba);
  return 0; $)
for(i=pocz;i<=kon;i++)$(
  if(*i=='(')$(
    poziom++;
    continue; $)
 if(*i==')')$(
    poziom--;
    continue; $)
  if(!poziom && (*i=='+' || *i=='-'))$(
    if(obl(pocz,i-1,liczba1)) return
-1;
    if(obl(i+1,kon,liczba2)) return -1;
    if(*i=='+') return
```

```
c_tadd(liczba1,liczba2,liczba);
      else return
c_tsub(liczba1,liczba2,liczba);
    $)
 $) /* for */
 if(poziom) return -1;
 for(i=pocz;i<=kon;i++)$(
   if(*i=='('$(
     poziom++;
     continue; $)

   if(*i==')'$(
     poziom--;
     continue; $)
   if(!poziom && (*i=='*' || *i==':'))$(
     if(obl(pocz,i-1,liczba1)) return
-1;
     if(obl(i+1,kon,liczba2)) return -1;
     if(*i=='*') return
c_tmul(liczba1,liczba2,liczba);
     else return
c_tdiv(liczba1,liczba2,liczba);
    $)
 $) /* for */
 if((*pocz=='(')&&(*kon==')')) return
obl(pocz+1,kon-1,liczba);
 return c_afp(pocz,liczba);
 $) /* koniec funkcji obl() */
```

Funkcja ryskon()

```
/*************************************
 *        FUNKCJA ryskon()           *
 *    rysuje kontur - ramke          *
 * argumenty funkcji:                *
 * x1,y1 - lewy gorny ros ramki      *
 * x2,y2 - prawy dolny ros ramki     *
 * rk    - rodzaj kreski (linii),    *
 *         ktora ma byc narysowana   *
 *         ramka                     *
 * rk ='c' linia cienka              *
 * rk ='g' linia gruba               *
 *************************************/

/* kody znakow uzytych do
 * kreslania konturu - ramki
 *      linia gruba
 *
 * lewy gorny ros          ATASCII 137
 * kreska pozioma gorna    ATASCII 149
 * kreska pozioma dolna    ATASCII 21
 * prawy gorny ros         ATASCII 143
 * kreska pionowa lewa     ATASCII 25
 * kreska pionowa prawa    ATASCII 153
 * lewy dolny ros          ATASCII 139
 * prawy dolny ros         ATASCII 140
 *
 *      linia cienka
 *
 * lewy gorny ros          ATASCII 17
 * kreska pozioma gorna    ATASCII 18
 * kreska pozioma dolna    ATASCII 18
 * prawy gorny ros         ATASCII 5
 * kreska pionowa lewa     ATASCII 124
 * kreska pionowa prawa    ATASCII 124
 * lewy dolny ros          ATASCII 26
 * prawy dolny ros         ATASCII 3
 */

ryskon(x1,y1,x2,y2,rk)
int x1,y1,x2,y2;
char rk;
$(
 int i;
 char lgr,kpg,kpd,pgr,kpl,kpp,ldr,pdr;
 switch(rk)
  $(
   case 'c' : lgr=17; /* kreska cienka
*/
              kpg=18;
              kpd=18;
              pgr=5;
              kpl=124;
              kpp=124;
              ldr=26;
              pdr=3;
              break;
   case 'g' : lgr=137; /* kreska gruba
*/
              kpg=149;
              kpd=21;
              pgr=143;
              kpl=25;
              kpp=153;
              ldr=139;
              pdr=140;
              break;
  $)    /* koniec switch */
 position(x1,y1);
 putchar(lgr);
 for(i=0;i<x2-x1-1;i++) putchar(kpg);
 putchar(pgr);
 for(i=y1+1;i<y2;i++)
  $(
   position(x1,i); putchar(kpl);
   position(x2,i); putchar(kpp);
  $) /* koniec petli for          */
 position(x1,y2);
 putchar(ldr);
 for(i=0;i<x2-x1-1;i++) putchar(kpd);
 putchar(pdr);
$)   /* koniec funkcji ryskon() */
```

Funkcja pisz()

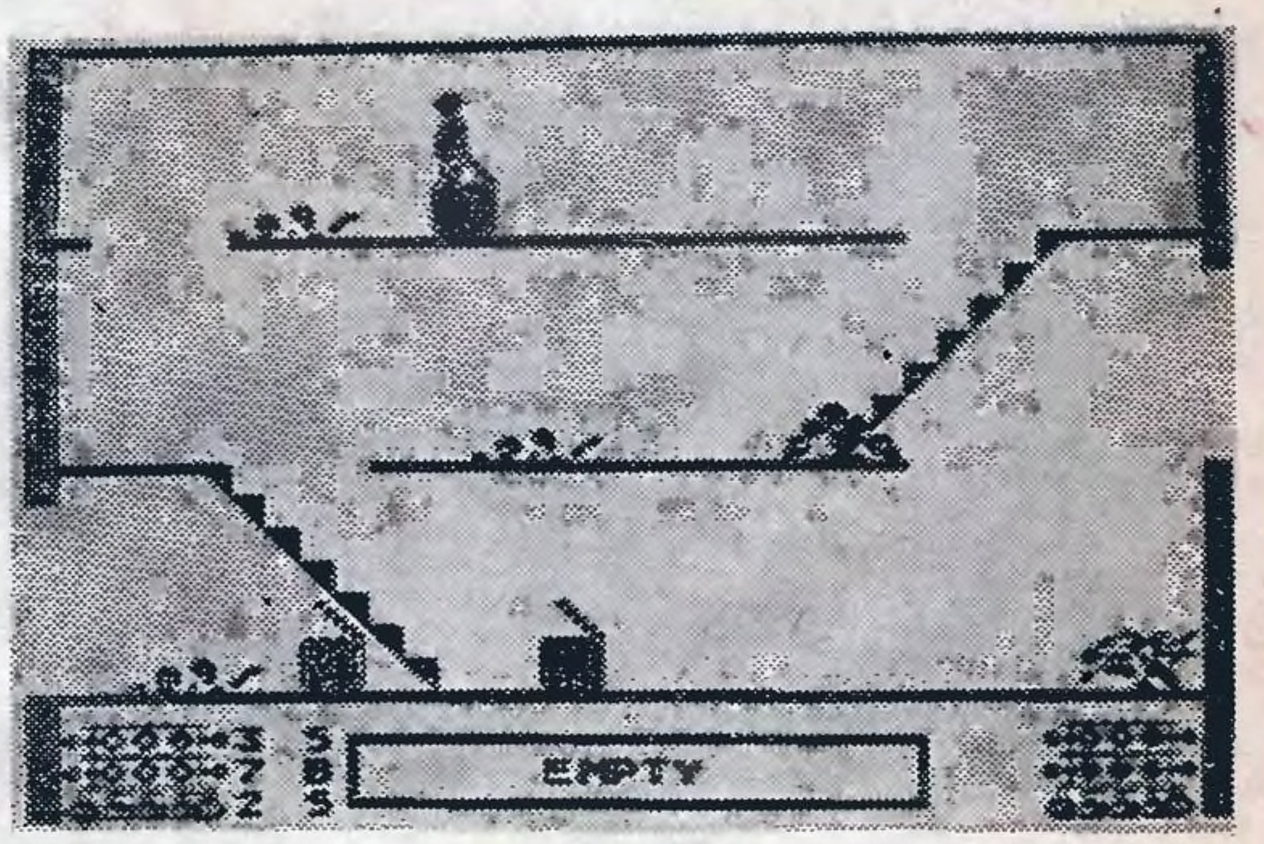
```
/*************************************
 *     FUNKCJA pisz()                *
 * wyswietla na ekranie lancuch zna- *
 * kow w miejscu okreslonym argumen- *
 * tami x,y.                         *
 * Ponadto jesli argument inv=128    *
 * to lancuch wyswietlany jest w     *
 * inwersie, inna wartosc inv nie    *
 * zmienia postaci wyswietlanego     *
 * lancucha.                         *
 * tekst - wyswietlany lancuch       *
 *************************************/

pisz(x,y,tekst,inv)
int x,y,inv;
char *tekst; /* wskaznik do lancucha */
$(
 char *kont; /* wskaznik do konca
                * lancucha                */
 int z;
 z=strlen(tekst);
 kont=tekst+z;
 position(x,y);
 for(;tekst<kont;tekst++)
  if(inv==128) putchar(*tekst|inv);
   else putchar(*tekst);
$)   /* koniec pisz */
```

AZTEC

Stara budowla z labiryntami tuneli ciagnących się bez końca to magnes przyciągający ludzi żądnych przygód , sławy a także bogactwa . Nie zważają na klątwy rzucone w zamierzchłych czasach przez kapłanów na intruzów , mogących zakłócić spokój prochom władców pochowanych w tych mauzoleach . Przykładów na to , że nie jest to znowu taka sobie fikcja , mamy wiele : ginący w niewyjaśnionych okolicznościach odkrywcy grobowców faraonów egipskich , uśmiercani przez dziwnych strażników awanturnicy , odnajdujący (w poszukiwaniu złota) miejsca kultu religijnego Indian amerykańskich , przypadki podobnych zdarzeń w Azji , Środkowej Afryce i Europie. Sądzę, że zaskoczeniem będzie sytuacja , jaka wytworzy się w momencie pojawienia się tego programu w Twoim mieszkaniu. Niebezpieczeństwo skarcenia Ciebie przez rodziców , np. z takiego powodu , że lekcje jeszcze nie odrobione lub kolacja już wystygła , jest znaczne . Do tego dochodzi również znaczne ryzyko " utraty życia " na skutek nieostrożności lub zbyt małej wprawy w poruszaniu się po podziemiach .

Po uruchomieniu programu ukaże Ci się menu . Dzięki niemu możesz urozmaicić i tak już bogatą w przygody grę . Początkowe litery określają , jaki klawisz należy nacisnąć aby uruchomić daną opcję . W przeciwieństwie do wielu programów (jeżeli oczywiście podołasz zadaniu) możesz wybrać wszystkie opcje jednocześnie . Nie zalecałbym jednak tego . Szczególnie przydatna opcja jest pierwsza (**ETERNAL STRENGTH**) . Dzięki niej stajesz się niezniszczalny . Aby przejść do następnej części programu należy nacisnąć klawisz **START** . Komputer zapyta się wtedy o stopień trudności w skali od 1 do 8 . Naciśnij któryś z tych klawiszy, a rozpocznie się przygoda . Na dole po lewej i prawej stronie ekranu umieszczone są Twoje zasoby i znaleziska : bomba (**STICKS**)- w momencie rozpoczecia gry masz ich trzy . Maksymalna ilość , jaka możesz ich zgromadzić, to siedem ; naboji (**BULLETS**) musisz , niestety szukać , zresztą tak samo jak pistoletu (**PISTOL**) , maczety (**MACHETE**) czy bożka (**THE IDOL**). Przy odrobinie szczęścia możesz znaleźć również dzban z życiodajną wodą , dzięki czemu powiększy się Twój zapas siły (**STRENGTH**) . Wszystkie te przedmioty ukryte będą w skrzyniach lub pod stertami śmieci . Aby sprawdzić czy one tam się znajdują , należy zatrzymać się , następnie nacisnąć przycisk raz , a wieko skrzyni otworzy się oraz ukaże się napis , co jest w środku , np. pusta (**EMPTY**) , czaszki (**SKULLS**) . Naciśnięcie jeszcze raz przycisku zapisuje przedmiot (jeżeli był) na Twoje konto . Sterte śmieci należy po prostu zebrać, naciskając przycisk manipulatora dotąd, aż miejsce nie oczyści się i pokaże czy coś się tam znajduje . Gdy tak będzie , naciśnij przycisk jeszcze raz , a staniesz się właścicielem tego przedmiotu . Jest, oczywiście , od tego odstępstwo, to znaczy bomba z zapalonym lontem . Zostaniesz ostrzeżony napisem (**DID WERE BLOWN UP !**) . Odsuń się wtedy na bezpieczną odległość i poczekaj aż nastąpi eksplozja .



Awanturnik , w którego się wcielasz może wykonywać szereg czynności. Wszystkie są wykonywane przy pomocy manipulatora. Poznanie i zapamiętanie uruchamiania poszczególnych czynności ułatwi Ci penetracje podziemii , a także uchroni przed czyhającymi niebezpieczeństwami . Oto niektóre z nich :

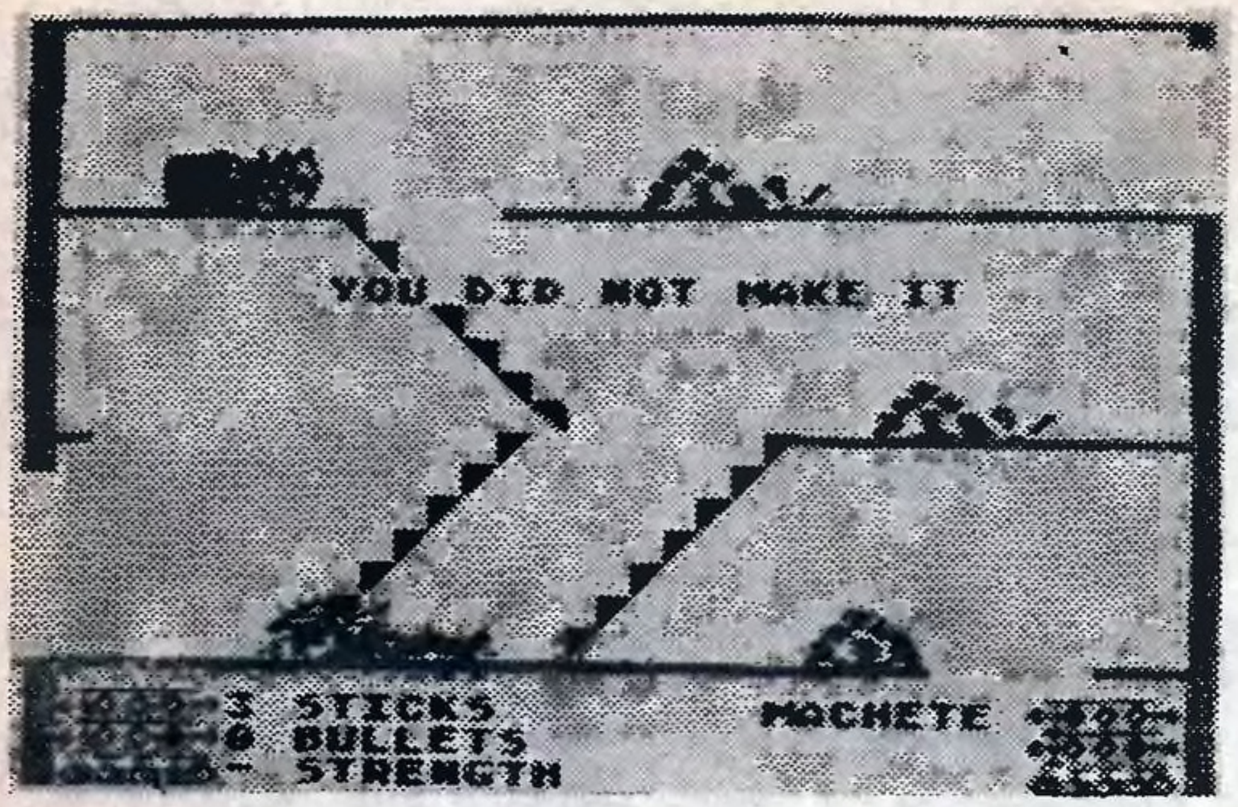
- Skierowanie manipulatora w lewo powoduje ruszenie w tę stronę postaci (kowboja) .
- Skierowanie manipulatora w prawo - kowboj rusza w prawo .
- Drążek manipulatora do góry - kowboj zatrzymuje się lub w przypadku , gdyby klęczał , wstaje .
- Drążek manipulatora do dołu - kowboj przykuca .
- Przutrzymanie drążka podczas poruszania się kowboja w lewo lub prawo sprawia , że zaczyna biec .
- Gdy kowboj idzie , a Ty naciśniesz przycisk manipulatora , zostanie przez niego wykonany skok .
- Gdy kowboj jest przykucnięty , a drążek skierujemy w lewo lub prawo , kowboj rozpocznie poruszać się na czworakach w dana strone .
- Na schody będziesz mógł wejść , gdy skierujesz drążek manipulatora pod kątem do góry .

We wszystkich swoich wędrówkach po korytarzach , gdy dopisze Ci szczęście , możesz znaleźć między innymi bombe , pistolet lub maczete . Przedmioty te bardzo się przydadza . Używa się ich w następujący sposób :

- Aby postawić bombe , kowboja należy ustawić w pozycji klęczącej z wyciągniętą ręka , przodem do miejsca , gdzie ma być podłożona bomba , a następnie nacisnąć przycisk manipulatora . Gdy pojawi się bomba z palącym się lontem , należy wyjść poza miejsce rażenia odłamkami bomby , ale pozostając w tym samym pomieszczeniu .

Stawianie bomby przydaje się do przebijania ścian lub posadzek w celu przedostania się do innego tunelu . Czasami mogą one uratować Ciebie od niechybnej śmierci , gdy wejdziesz do tunelu, z którego nie ma wyjścia .

- Gdy masz pistolet (bo znalazłeś go np. w skrzyni) i chcesz go użyć , sprawdź w pierwszej kolejności , czy posiadasz do niego naboje (**BULLETS**) . Informacje o



nich masz w lewej dolnej części ekranu . Skierowanie manipulatora do góry i jednoczesne naciśnięcie przycisku sprawi, że kowboj stanie w pozycji gotowej do oddania strzału .

- Oddanie strzału odbywa się przez naciśnięcie przycisku manipulatora .

- Aby schować pistolet , należy skierować drążek manipulatora do góry oraz nacisnąć przycisk . Z wyciągniętą bronią kowboj może poruszać się tylko w poziomie .

- Gdy masz maczetę i chcesz jej użyć , skieruj drążek manipulatora do góry i naciśnij jednocześnie przycisk. Bronią tą możesz zadawać dwa rodzaje ciosów . Pierwszy to pchnięcie na wprost z wysunięciem nogi, wygląda to jak atak szpada . Wykonuje się go przez naciskanie przycisku. Drugi to cięcie obronne w dół , szczególnie przydatny podczas ataku małych zwierząt ; należy w tym przypadku skierować uchwyt manipulatora do siebie .

- W przypadku , gdy posiadasz zarówno pistolet jak i maczetę , do walki możesz używać jednej z tych broni . Gdy walczysz maczetą i w pewnej chwili dochodzisz do wniosku, że lepiej użyć jest pistoletu, skieruj drążek manipulatora do góry , a nastąpi zmiana broni .

W czasie przeszukiwania korytarzy napotkasz wiele groźnych zwierząt ; lwa , zwierzę prehistoryczne , ośmiornicę , a także Indianina będziesz mógł " uspokoić ", strzelając do nich z pistoletu . Kobrę i tygrysa możesz uśmiercić z pistoletu , jak również siekąc maczetą . Pozostałych zwierząt możesz pozbyć się przy pomocy maczety . W przypadku , gdy nie dysponujesz jeszcze żadną bronią , jedyną radą jest ucieczka lub jeżeli jest to : wąż , skorpion lub mały pająk , możesz po prostu je przeskoczyć . Długość korytarzy i liczba ich poziomów jest ograniczona . Stąd i ilość atrakcji ma swoją granicę . Czas, który uda Ci się w nich spędzić, wystarczy aż nadto , aby zaspokoić Twoją potrzebę wrażeń i przyjaciół . Gdy ukaże się napis **YOU DID NOT MAKE IT** , będzie oznaczało , że już nic w tej grze nie zrobisz, a więc możesz wcisnąć klawisz **START** i rozpocząć grę od nowa .

DIMENSION X

Nie będę rozpisywał się , jak to się stało , że na Twą nieduża planetę o nazwie Jaraloba "zawitali" wrogowie . Skoro jednak tak się stało , siadaj za sterami . Nim wystartujesz , przypomnij sobie kilka istotnych informacji .

Po pierwsze : które klawisze oraz do czego są używane w grze . Wierzę , że odpowiesz następująco :
OPTION -uzyskanie na ekranie menu;
SELECT - wybór opcji w menu ;
START - start programu ;
HELP - przejście od gry do menu;
SPACE - zatrzymanie gry . Powtórzenie tego klawisza ponownie uruchamia program (kontynuowanie gry) .

Po drugie : tablica przyrządów zawiera szereg wskaźników . Wszystkie są ważne. Rozmieszczenie ich przedstawia poniższy rysunek .

TABLICA PRZYRZĄDÓW :

1 -Tablica komunikatów komputera pokładowego ;
2 -Informacja o ilości ekranów ochronnych ;
3-Wskaźnik ilości energii statku;
4 -Wskaźnik odległości od nie - przyjaciela ;
5 - Ekran radaru ;
6 - Mapa sektorów planety .

W celu ułatwienia posługiwania się poszczególnymi wskaźnikami zostaną one kolejno omówione .

Podczas lotu należy zwracać uwagę na tablicę komunikatów . Ukazujące się na niej informacje w istotny sposób mogą wpłynąć na całokształt gry, jeżeli będziesz się do nich dostosowywał . A oto ich wykaz wraz z objaśnieniami :

- AUTOMATIC BOOSTERS ENGAGED - automaty włączyły silniki biplazmowe. Komunikat ten ma miejsce tylko podczas przechodzenia statku przez sektory , w których nic nie występuje . Ma to na celu szybsze przejście przez sektory . W czasie działania tych silników wyłączone są ekrany ochronne .
- BOOSTERS DISENGAGED - nastąpiło wyłączenie silników biplazmowych .

Można do tego doprowadzić przez poruszanie drążka manipulatora .

- DESORT MAP HIT - nastąpiło uszkodzenie mapy .
- EDGE OF SECTOR AHEAD - pojazd zbliża się do końca sektora . Równocześnie z tym komunikatem uruchamiany jest sygnał dźwiękowy.
- END OF TUNNEL AHEAD - statek zbliża się do końca lotu przez dziesiąty wymiar . Tak jak w przypadku zbliżania się do końca sektora , tak i tu dodatkowo informowani jesteśmy sygnałem dźwiękowym .
- FUEL TANKS HIT - przeciwnik usz-

kodził zbiorniki paliwa . Paliwo wydobtaje się z nich , co grozi katastrofa .

- LOW ON FUEL -zaczyna wyczerpywać się paliwo . Dodatkowym ostrzeżeniem jest sygnał dźwiękowy .
RIGILLAN APPROACHING - do statku zbliza się nieprzyjaciel .
- SCANNER HIT - nastąpiło uszkodzenie , a tym samym unieruchomienie głównego czujnika (scannera) .
- SECTOR EMPTY - komunikat ten informuje , że w sektorze , w którym aktualnie się znajdujesz , nie ma nic .
- SECTOR SECURED - chwilowo w sektorze jest bezpiecznie , mimo że znajdują się w nim statki przeciwnika .
- SHIELD DIENTEGRATING -ze względu na trafienia osłabiona jest moc ekranu ochronnego .
- THE CAPITOL IS SURROUNDED - wróg otoczył stolicę . Jeżeli go teraz nie zniszczysz , wedrze się do miasta , a tym samym będzie to oznaczało Twój koniec .
- WELCOME ABOARD WARRIOR - komunikat ten występuje na początku gry. Jest to grzecznościowy zwrot powitania na pokładzie statku .
- ZONE DEFLECTION - Twój statek znajduje się w strefie neutralnej. Tracisz jeden poziom ekranu ochronnego .

Wskaźnik informujący o ilości ekranów ochronnych da Ci początkowo luksus wzrokowego bezpieczeństwa , lecz w miarę , jak będziesz inkasował celne strzały wroga , spojrzenia kierowane na ten element tablicy będą coraz bardziej nerwowe .

Odporność uderzeniowa ekranów ochronnych dzieli się na trzy grupy :
- silne (STRONG) wytrzymujące trzy trafienia z działa przeciwnika ;
- normalne (NORMAL) wytrzymujące dwa trafienia z działa ;
- słabe (WEAK) wytrzymujące tylko jedno trafienie z działa przeciwnika .

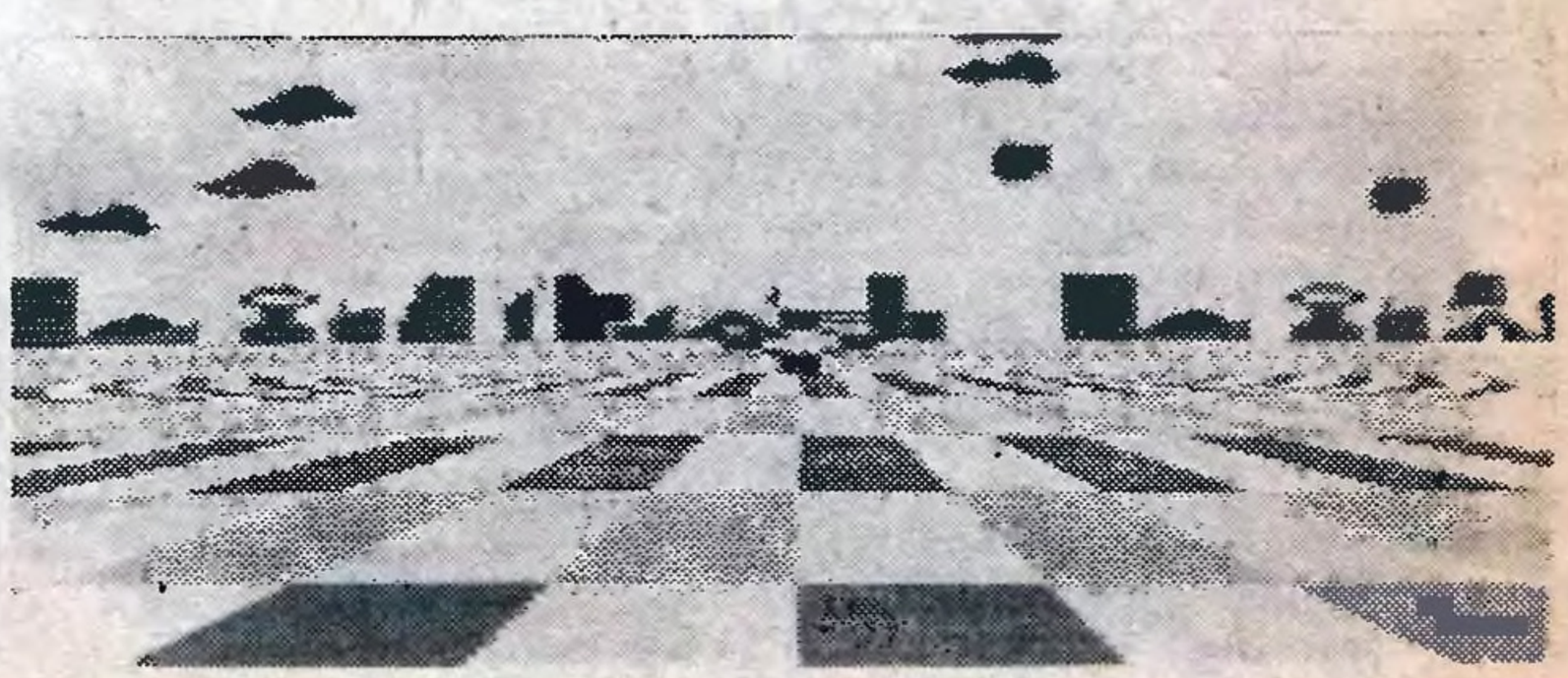
Rozpoczynając grę , dysponujesz pięcioma ekranami : zielonym , żółtym , czerwonym , krytycznym i zerowym . W trakcie gry , gdy otrzymujesz trafienia od przeciwnika , przechodzisz na coraz niższe poziomy ekranów ochronnych. Aktualną sytuację otrzymujesz na tablicy przyrządów .

Energia Twojego statku w miarę lotu powoli wyczerpuje się . Będziesz mógł to obserwować na wskaźniku ilości energii statku . Gdy zauważysz , że zapas jej wyczerpuje się , odszukaj na mapie sektor, w którym znajduje się baza paliwa . Spróbuj do niej dotrzeć . Gdy uda Ci się dobrnąć do sektora, w którym ona się znajduje , pojawi się na ekranie cały szereg informacji . Komputer kolejno sprawdza zespoły statku , jeżeli wykryje uszkodzenie , naprawi je , o czym zostaniesz powiadomiony . Gdy statek zostanie doprowadzony do pełnej zdolności bojowej , rozpoczyna się odliczanie czasu , po upływie którego statek Twój zostanie wystrzelony do migającego pomarańczowego sektora. Jeżeli chcesz być przeniesiony w inny niż wskazany sektor , ustaw jego nowe położenie przy pomocy manipulatora . Musisz jednak zdążyć z tą czynnością w czasie odliczania .

W czasie lotu przez sektor możesz napotkać statki wroga . Zniszczenie ich jest Twoim obowiązkiem. Przed dojściem do styczności wzrokowej znasz odległość do nieprzyjaciela dzięki wskaźnikowi odległości . Odległość do wroga jest o tyle istotna , że możesz dokonać manewru wyprzedzającego ich uderzenie .
Liczba ujemna na liczniku informuje, że statki nieprzyjaciela znajdują się za Tobą .

Podobną rolę do wskaźnika odległości pełni radar pokładowy (QUADRA SCANNER).
Obraz na monitorze radaru pokazuje rozmieszczenie statków przeciwnika (błyskające białe kropki) w stosunku do Twojego pojazdu (żółta kropka).

Podstawowej informacji na temat położenia Twojego statku na planecie udziela mapa sektorów. Składa się ona z 25 pól (5x5) . Wyświetlane są na niej informacje ważne dla dalszych Twoich poczynań . Pooznaczane są różnymi kolorami , literami i cyframi .
- Pole koloru **pomarańczowego szybko pulsujące** z czarną ramką - oznacza sektor, w którym właśnie się znajdujesz .
- Pole koloru **pomarańczowego wolno pulsujące** - wskazuje sektor , do którego zostaniesz przeniesiony .



- oznacza sektor pus-

...leskie - jest wskazni-
...ra , ktory do tej pory
...stał jeszcze zbadany . Stad
... o nim jakichś konkretnych
...domości .

Pole białe w kropki -przedstawia sobą strefę neutralną .

- Litera C na polu - ozn. sektor , w którym znajduje się stolica .
- Litera F na polu - określa miejsce lokalizacji bazy paliwowej dla Twego statku .

Cyfra na polu - informuje , ile statków nieprzyjaciela znajduje się w danym sektorze .

Stopień utrudnienia gry możesz sam ustalić tak , byś nie zniechęcił się do niej po paru minutach zabawy . Czynności tych powinieneś dokonać przed uruchomieniem gry . Służy do tego menu , w którym występują trzy opcje :

- Wyszkolenie pilota (PILOT SKILL) . Dzieli się na : nowicjusza (novice) , zaawansowany (advanced) i mistrz (expert) ;
- Odporność uderzeniowa ekranów ochronnych (SHIELD STRENGTH) została omówiona wcześniej ;
- Liczba przeciwników w sektorze (ALIENS PER SECTOR) . Mamy możliwość wyboru jednego z trzech wariantów : niewielu (few), normalna liczba (normal) i wielu (many) .

Szczególnie w pierwszych podejściach do gry nie za bardzo będziesz zadowolony ze swoich umiejętności . Niech Cię to nie zraża , spróbuj jeszcze raz .

Gdy nie uda Ci się przebyć przez dziesiąty wymiar (rozbijesz się) lub zastrzeli Cię wróg , otrzymasz na ten temat stosowny komunikat . Dowiesz się z niego między innymi o tym , że planeta Jaraloba , tc jest ta , której broniłeś została opanowana przez Rigillians, niestety , z Twojej winy . Poinformowany zostaniesz również o ocenie, jaką wystawił Ci za całość zmagań komputer np. NOVICE WARRIOR CLASS 2 .

SPY vs SPY

Wyścig zbrojeń trwa i taki stan będzie istniał , dopóki zło nie zostanie wyplenione z myśli i czynów ludzi. Powtarzanie tego zdania nie robi już wrażenia , gdyż jest ono prawdą powtarzaną od wieków . Przytaczam je dlatego , że będzie ono po części dotyczyło i Ciebie . Ministerstwo do spraw wojskowych w Twojej ojczyźnie prowadzi doświadczenia z nowym typem rakiet międzykontynentalnych . Jest to odpowiedź na takie same doświadczenia prowadzone przez państwo , mające złe zamiary w stosunku do Twej ojczyzny . Jak zwykle zdarzają się próby nieudane . Tak właśnie się stało . Rakieta po przelocie nad oceanem kilkuset kilometrów nieoczekiwanie spadła na niewielką tropikalną wyspę należącą do Archipelagu Smoka, rozpadając się na trzy części . Natychmiast zostały podjęte środki mające na celu odzyskanie rakiety , a także uniemożliwienie dostępu do niej szpiegów wroga. Jako fachowiec wysokiej klasy otrzymujesz zadanie odzyskania rakiety . W tym celu zostajesz przetransportowany na archipelag . Na wyspę dostajesz się , zostając zrzuconym na spadochronie. Jednocześnie w kierunku wyspy została wysłana łódź podwodna do odebrania rakiety , oczywiście po znalezieniu i zmontowaniu wszystkich części przez Ciebie . Przeciwnik też nie śpi . Wysłał na wyspę szpiega , zrzucając go na spadochronie z takim samym zadaniem , z jakim Ty się tam znalazłeś . Obaj jesteście wyposażeni w najnowocześniejszy sprzęt , jaki wyprodukowano dla prowadzenia działalności szpiegowskiej . Będziecie walczyć ze sobą oraz z czasem , gdyż na wyspie znajduje się wybuchający co pewien czas wulkan . Wybuch jego zniszczy wyspę , a więc uniemożliwi wykonanie zadania .

Po uruchomieniu programu zobaczysz po prawej stronie zestaw opcji :

- Zegar podający czas do erupcji wulkanu ;
- Liczba graczy (PLAYERS) 1 lub 2;
- Poziom trudności (LEVEL) od 1 do 7 ;
- Stopień "inteligencji" komputera (I.Q.) od 1 do 5 ;
- Ujawnienie łodzi podwodnej tak (Y) lub nie (N) .

Opcje wybiera się przez ruch manipulatora do lub od siebie. Wybrana opcja pulsuje . Poziom trudności wybiera się przez skierowanie manipulatora w lewo lub prawo . Gdy uznasz , że wszystkie elementy ustawiłeś właściwie , naciśnij przycisk manipulatora lub klawisz spacji i będziesz mógł rozpocząć grę .

W czasie gry będziesz mógł, a nieraz i musiał , używać niektórych klawiszy :

A - ruch w górę lub w dołu ;
Z - ruch do dołu lub do przodu ;
K - ruch w lewo ;
L - ruch w prawo ;

@ - włączenie urządzenia przenoszącego w inny wymiar ;
S - włączenie lub wyłączenie muzyki ;
ESC - zatrzymanie programu oraz powtórne jego uruchomienie ;
SPACE - rozpoczęcie gry , podniesienie lub opuszczenie przedmiotu, wybór i odbezpieczenie miny ;
RESET -ponowne ładowanie programu;

OPTION - powrót programu do ekranu z opcjami .

Niektórych klawiszy używa się również przy wciśniętym jednocześnie klawiszu CONTROL , a mianowicie : (podczas zadawania ciosów szablą)
Klawisz A - podniesienie ręki z szablą do góry ;
Klawisz K - ręka do siebie ;
Klawisz L - pchnięcie szablą ;
Klawisz Z - cięcie szablą .

Bardzo ważnym przedmiotem w grze jest urządzenie , przy pomocy którego możesz przechowywać przedmioty znalezione na wyspie . Działa ono na zasadzie przenoszenia w inny wymiar i z powrotem . W ten sposób możesz gromadzić wszystko , co przedstawiają symbole po lewej stronie okna . Przenoszenie w inny wymiar odbywa się za pomocą przycisku manipulatora . Pierwsze przyciśnięcie powoduje podniesienie przedmiotu , drugie przeniesienie do innego wymiaru. Obecność w nim tego przedmiotu sygnalizowana jest zapalonym wskaźnikiem obok symbolu . Użyć któregoś z przed-

niotów można również przez naciśnięcie przycisku manipulatora , następnie wybranie przez ruch drążkiem do dołu lub góry i ponownym naciśnięciu jego przycisku . Szczególnym przedmiotem w tym wykazie jest mapa wyspy . Poruszając się po wyspie w poszukiwaniu części rakiety , możemy w każdej chwili skorzystać z tej mapy , aby zlokalizować ich miejsca . Mapa będzie dzieliła wyspę na sektory . Ilość sektorów , których może być do 22, będzie zależna od wielkości wyspy , a więc od stopnia trudności. Części rakiety będą widoczne na wyspie w postaci białych kwadracików . Będzie również zaznaczona twoja droga .

Części rakiety będą leżały na wyspie ukryte pod pryzmami piasku, wyglądającymi każda jak trzy małe trójkąty. Dopiero ich podniesienie ujawni nam przedmiot , jaki znajduje się pod nimi . Nieraz może to być przykre, gdyż może okazać się, że odkryty przedmiot to mina , która za chwilę wybuchnie . Warto dlatego zaglądać do mapy .
Również czas skłania do tego, gdyż na odnalezienie i wywiezienie rakiety jest go niewiele . W zależności od stopnia trudności czas ten do wybuchu wulkanu wynosi (licząc od stopnia najłatwiejszego) 6, 8, 11, 14, 18, 22, 27 minut. Ze stopniem trudności związanych jest również wiele zmian , gdyż ma się do czynienia z coraz większą wyspą, ilością przedmiotów na wyspie , a także ilością wysp .

Poruszanie się po wyspie można, oczywiście , realizować także za pomocą manipulatora. Gdy dojdziesz do lewego lub prawego skraju ekranu , następuje jego zmiana (pokazany jest kolejny sektor wyspy). Inaczej ma się sprawa z przechodzeniem sektorów w górę lub dół . Nie w każdym sektorze możesz to zrobić . Umożliwią Ci to sektory , w których będą większe luki między drzewami lub przerwa w dolnej części okna .

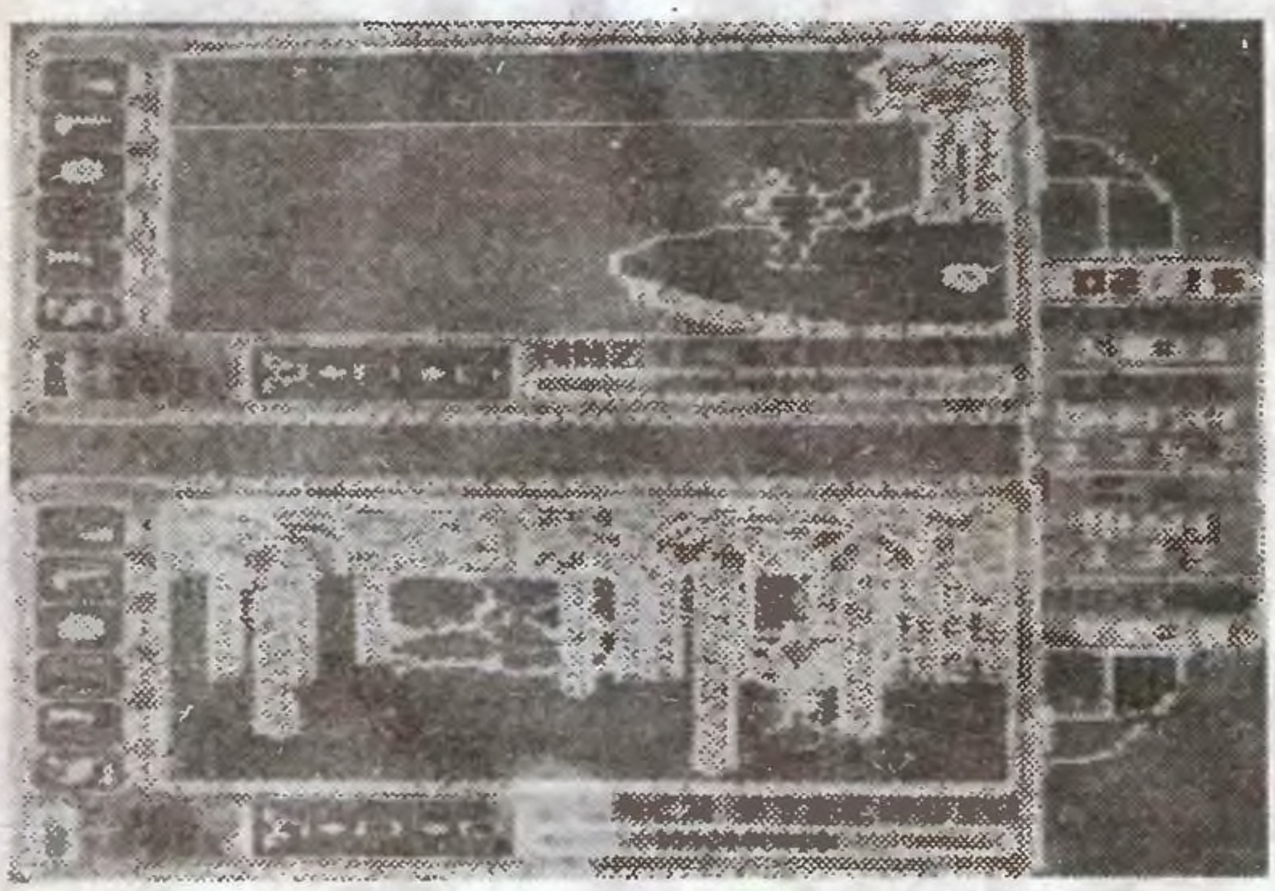
Na wyspie czyhą na Ciebie szereg pułapek : wymienione wcześniej zranienie przez granat , można wpaść w bagno (miejsca o ciemniejszym tle) . Wydostać możesz się z niego przez szybki pionowy ruch manipulatora lub kręcenie nim w kółko . Czeka Cię jednak pewna utrata czasu .
Możesz wpaść również w pułapkę , którą sam założyłeś .
Gdy złożysz całą rakietę , przyjdzie kolej na przetransportowanie jej na łódź podwodną . Na brzegu nie ma łódki , musisz zrobić to wpław . Pływanie zabiera Ci dużo [illegible] odpływaj od brzegu w [illegible] okrętu , gdyż możesz [illegible] Drugie niebezpieczeństwo [illegible] może spotkać Ciebie w wodzie [illegible] rekiny . Szybko musisz wtedy [illegible] nać , by ujść ich szczękom .

Wojowniczość szpiega jest [illegible] tępnym niebezpieczeństwem . Gdy znajdzie się w tym samym sektorze wyspy , może zaatakować , strzelając do Ciebie z pistoletu lub siekąc szablą . Stopień zmęczenia masz ukazany po prawej stronie pod oknem . Twoje zmęczenia to **WHILE STRENGTH**, a wroga **BLACK STRENGTH** . Przy odrobinie szczęścia możesz szpiega dzięki inicjatywie , uporowi i zręczności wystać na tamten świat , a następnie postawić mu pomnik .
Gra umożliwia Ci zastawianie pułapek na przeciwnika . Służą do tego następujące przedmioty : lina , łopata , napalm , bomba . W zależności od stopnia trudności niektóre z tych przedmiotów możesz mieć ze sobą dzięki urządzeniu przenoszącemu w inny wymiar , inne dopiero będziesz mógł znaleźć na wyspie . Metoda wydobywania ich z innego wymiaru oraz sposób szukania na wyspie zostały omówione wcześniej .

Sposób zastawiania pułapek jest prosty .

Lina jest wykorzystywana do wykonania potrząsku . Należy wejść na drzewo , zaczepić jeden koniec liny . Następnie zejść z drzewa i zaczepić ją w dowolnym miejscu na ziemi (nacisnąć przycisk) , lina stanie się niewidoczna .
Ukrywanie bomb oraz napalmu jest jeszcze prostrze . Po położeniu na ziemi upodabniają się one do górek z piasku (trzy trójkąty) .
Przy pomocy łopaty możesz wykopać szereg dołów .
Pułapki te są groźne dla wroga , a także i dla Ciebie , a więc pamiętaj o tym , gdzie je zastawiłeś .

W trakcie wędrówki po wyspie tracisz ogólną energię . Jedyną



" IKS " - dodatek " Żołnierza Wolności ". Redaguje Wiesław Cetera (kierownik zespołu), Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski , Ireneusz Miernik , Ludwik Piela , Jacek Szaniawski . Adres redakcji : 00-950 Warszawa ul . Grzybowska 77 , telefon centrali 20-12-61 w. 486. Telex 313664 .

Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów . Nakładem Wydawnictwa " Czasopisma Wojskowe ", Warszawa ul. Grzybowska 77. Druk offsetowy - Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego.

Nr Zam. 1981 Nr ind. 396281. U-28