UNIWERSYTET
JAGIELLOŃSKI
W KRAKOWIE

Wprowadzenie do Języka Atari BASIC

Szkoła Letnia Twórczego Programowania 2018

mgr inż. Tomasz D. Cieślewicz

Podstawy Atari BASIC

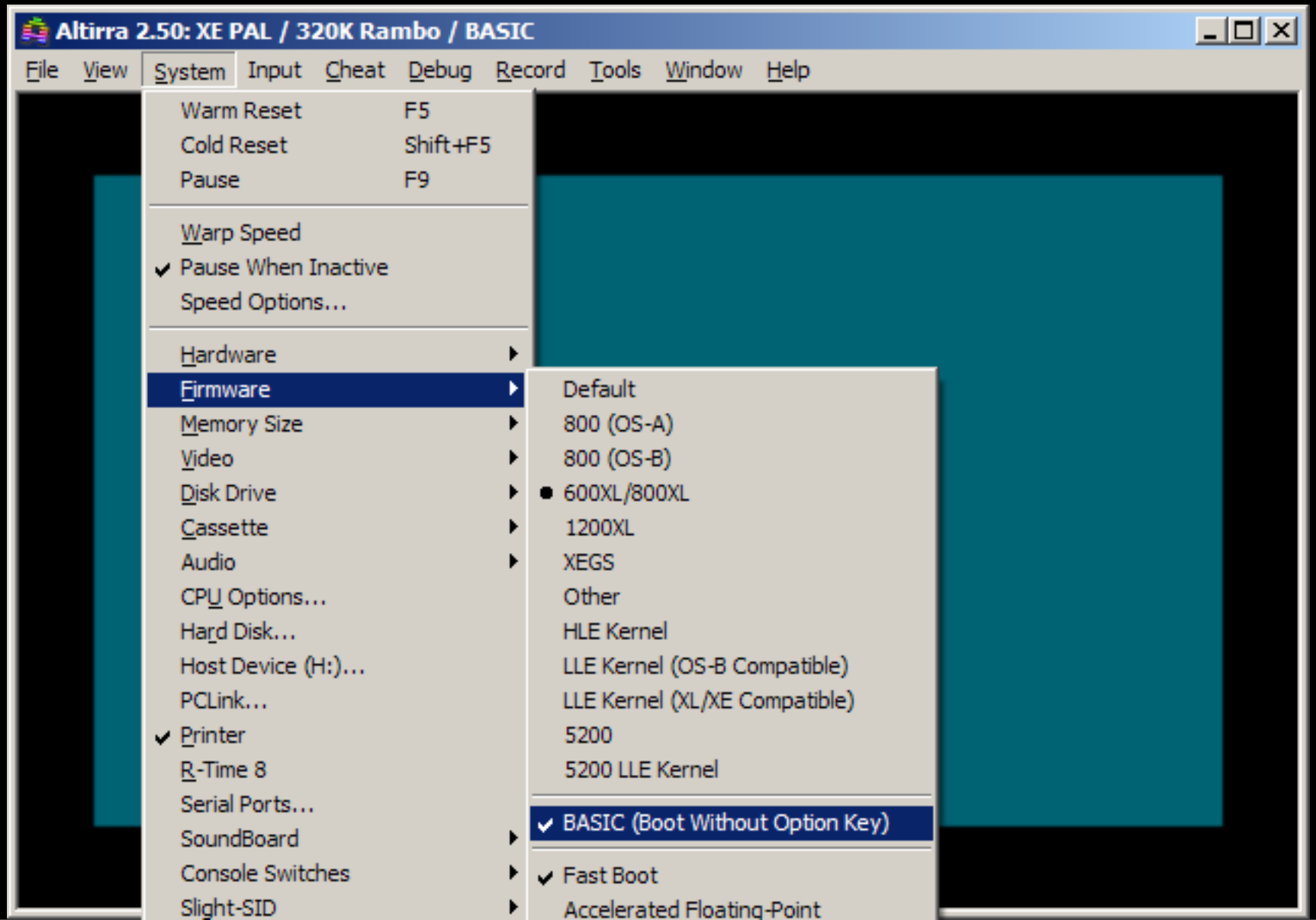
(Beginners All-purpose Symbolic Instruction Code)

wersja dla Atari powstała w 1978 roku

Język BASIC uruchamia się samoczynnie po włączeniu komputera (większość komputerów 8-bitowych posiada taki język wbudowany).

Jeśli język się nie uruchomi to być może komputer jest uruchomiony z włączonym klawiszem Option.

Klawisz Option w emulatorze:



Wprowadzane komendy są od razu wykonywane, np.

? wyrażenie

Wyświetla wartość wyrażania

(lub wyświetla wartość zmiennej/zmiennych)

Wprowadzenie komendy z numerem linii zapamięta ją (nie będzie wykonana, ale zostanie sprawdzona poprawność składni), np.

10 ? ”Witaj!

Program w pamięci uruchamia się komendą

RUN

Wykonana zostanie pierwsza linia bez względu jaki ma numer.

Następnie będą wykonywane kolejne linie (rosnąco), bez względu na to jaki będzie ich numer (ściśle odstęp numeracji między nimi, często jest to 10 numerów np. 10, 20, 30 itp.)

Numeracja
linii od 0
do 32767

LIST – wyświetla kod programu w pamięci

L. 10 – wyświetli linię o numerze 10

L. 30,100 – wyświetli linie o numerach z przedziału

LIST

Jeśli program nie mieści się na ekranie, klawisze:

CONTROL+1 zatrzymują wyświetlanie, które można następie kontynuować (ponownie naciskając CONTROL+1).

Klawisz Break (Pause na pececie w emulatorze) przerywa listowanie programu i umożliwia jego dalszą edycję.

Sterowanie programem – podsumowanie

NEW – kasuje pamięć BASIC (twój program)

END – kończy działanie programu

RUN – uruchom program (skrót RU.)

LIST – wyświetla kod programu (skrót L.)

Klawisz Break (Pause na pececie) przerywa działanie programu

CONT – wznawia przerwany program

Zapisywanie i wczytywanie programu z dyskietki:

1. Tryb tekstowy

LIST „D:gra1.txt” - zapis kodu jako tekst do pliku

gra1.txt

1 jest numerem twojej grupy

ENTER „D:gra1.txt” - odczyt kodu programu jako tekst

2. Tryb natywny

LOAD „D:gra1.bas” i SAVE „D:gra1.bas”

Podstawowe komendy języka Atari BASIC:

PRINT – pisz (skróty: PR. lub ?)

POSITION x, y – ustawia kursor w pozycji x, y (skrót POS.)

INPUT zmienna – wprowadź wartość zmiennej z klawiatury (skrót I.)

FOR zmienna=wart_startowa TO wart_koncowa / STEP n : ... : next zmienna

GOTO numer linii – skok do linii

IF warunek THEN ... – wykonaj jeśli warunek jest spełniony

END – koniec programu (często nieobowiązkowe)

RESTORE + READ + DATA – baza danych

Zmienne

Zmienne numeryczne nie wymagają deklarowania (deklaracja odbywa się automatycznie przy pierwszym użyciu zmiennej, wartość zmiennej jest równa zero). Np.

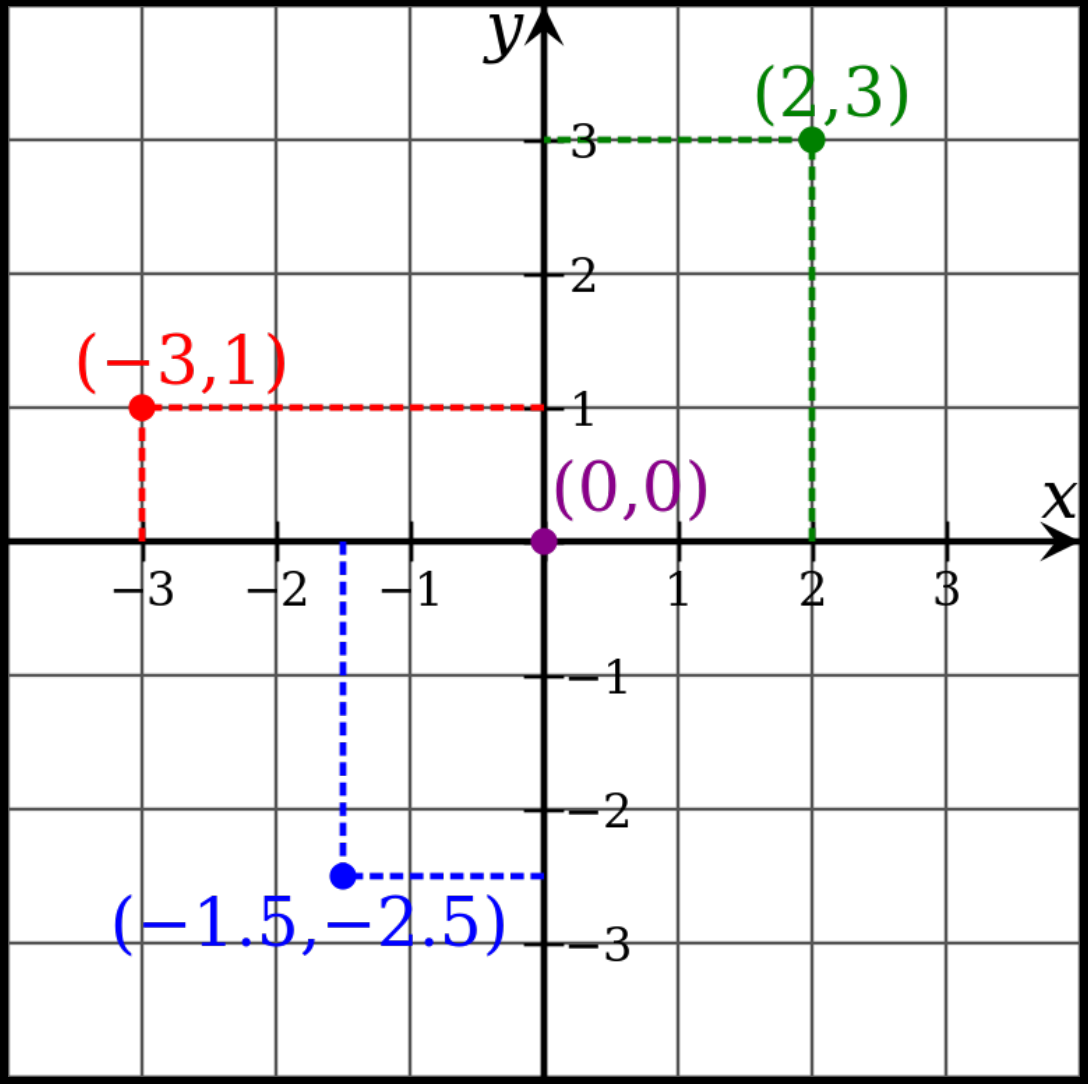
100 Zmienna=5

Zmienne tekstowe (string), wymagają deklarowania jako tablice. Np.

10 DIM A$(200)

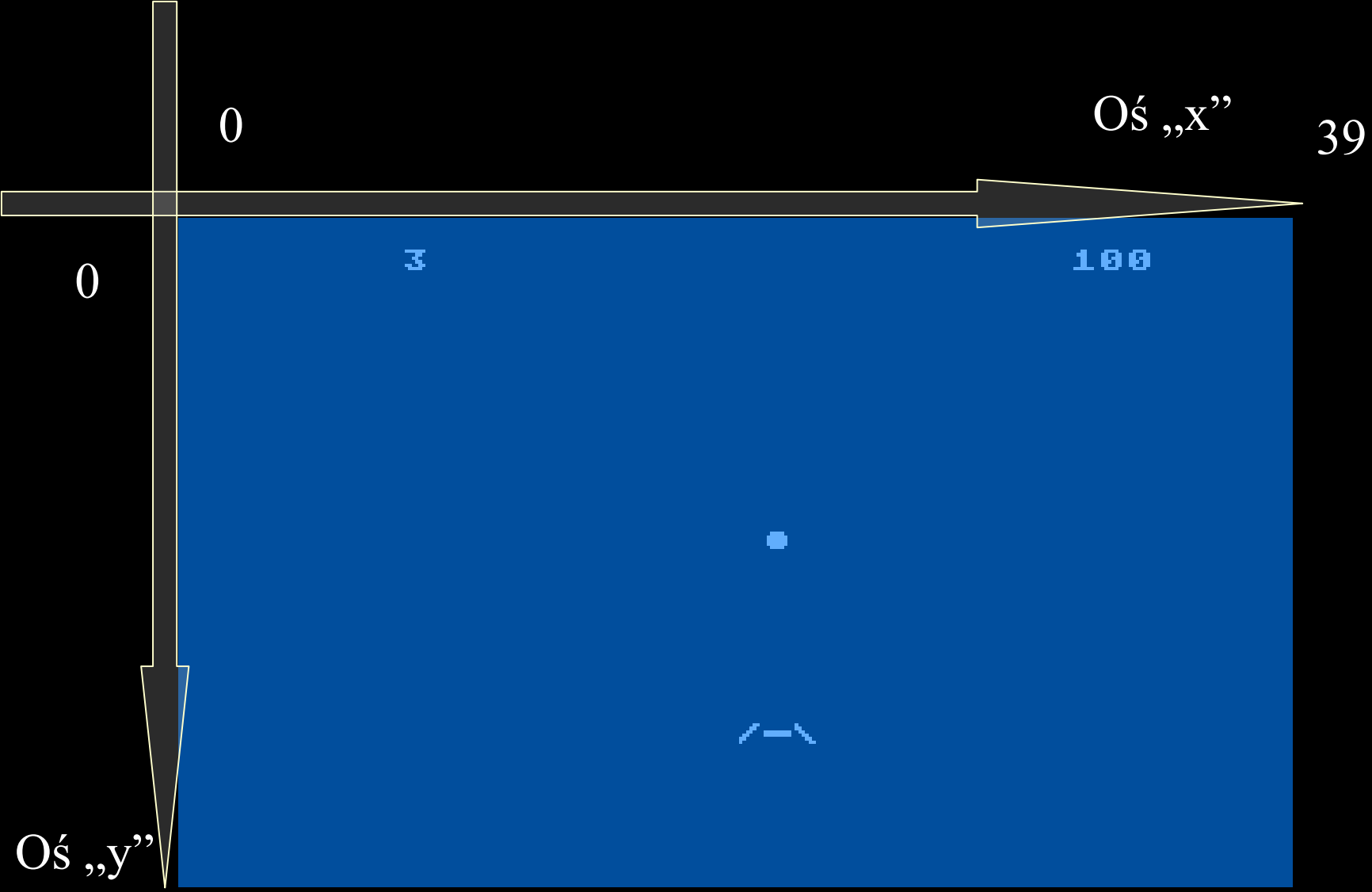
gdzie 200 oznacza ilość bajtów (max rozmiar tekstu)

Position

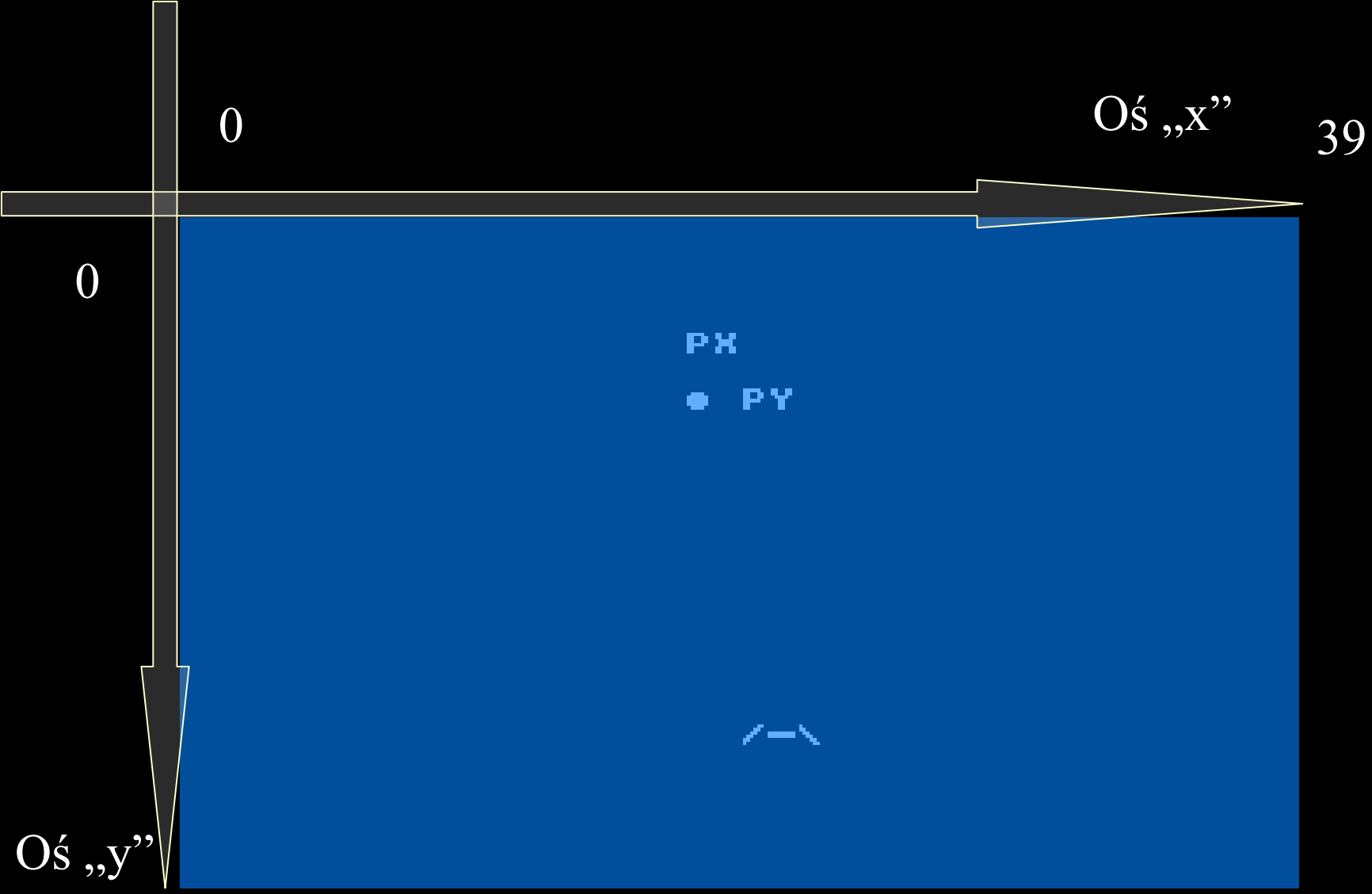


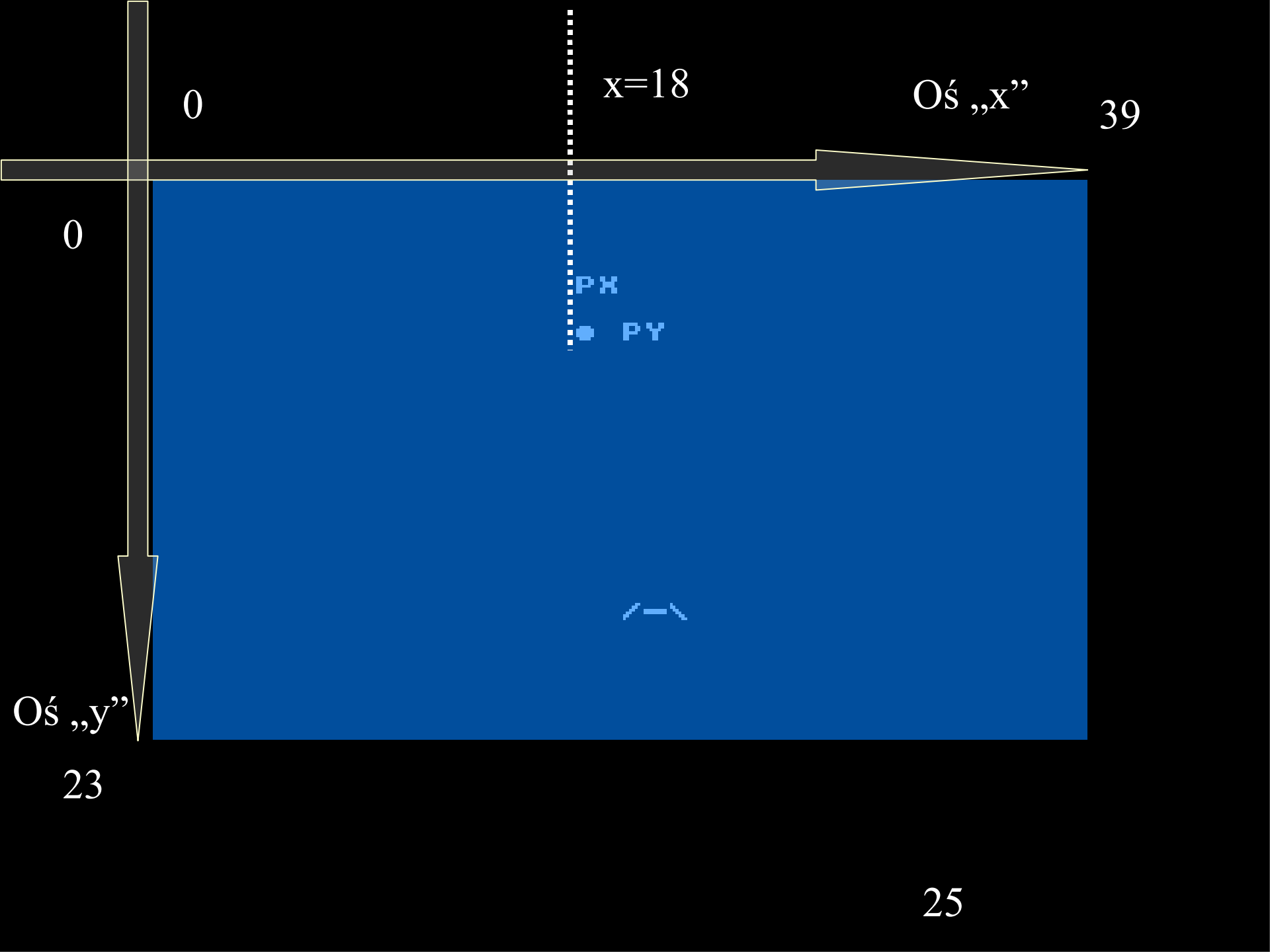
Układ współrzędnych kartezjańskich

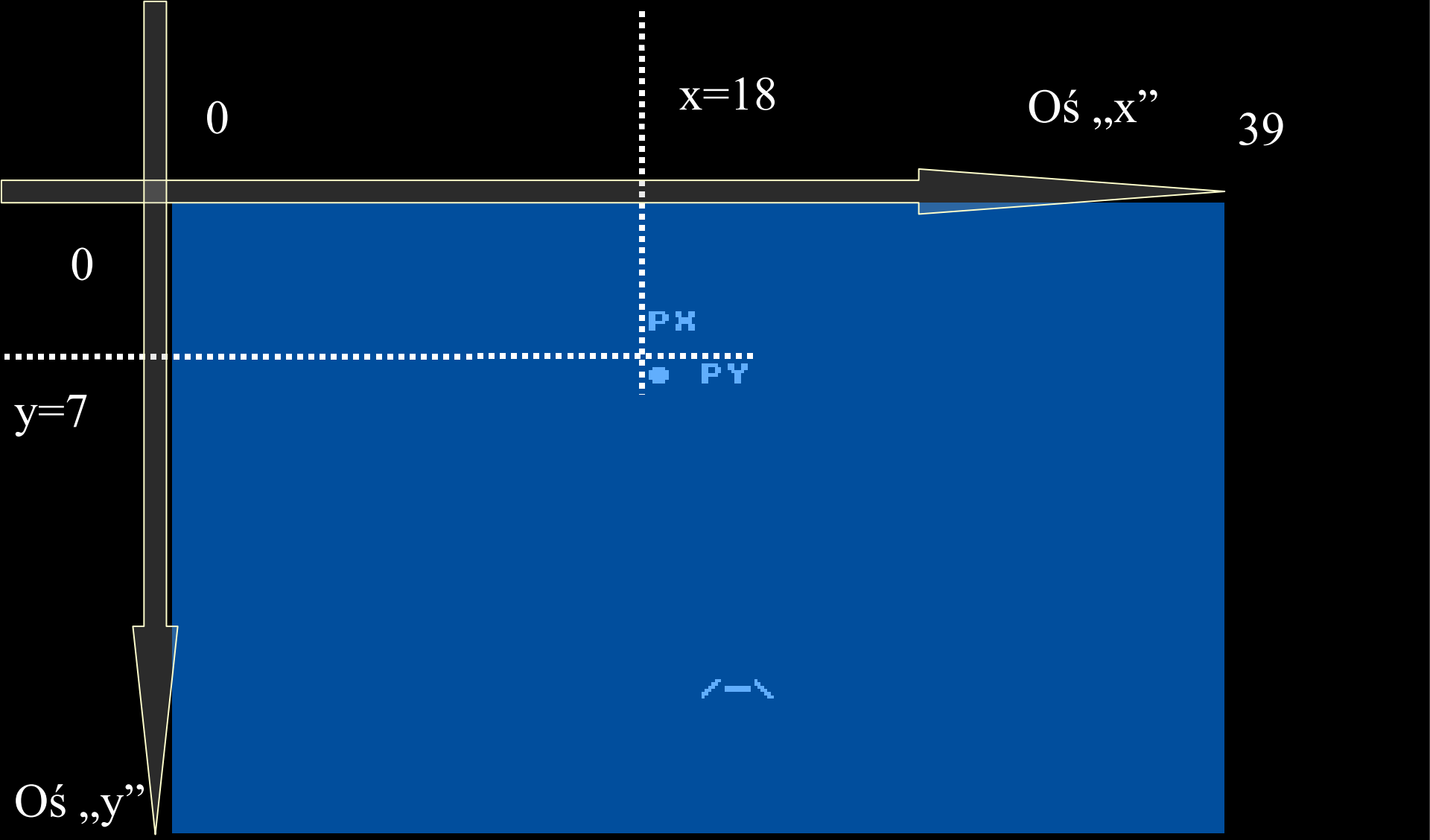
Źr. grafiki pl.wikipedia.org



Ekran Atari w trybie tekstowym 0

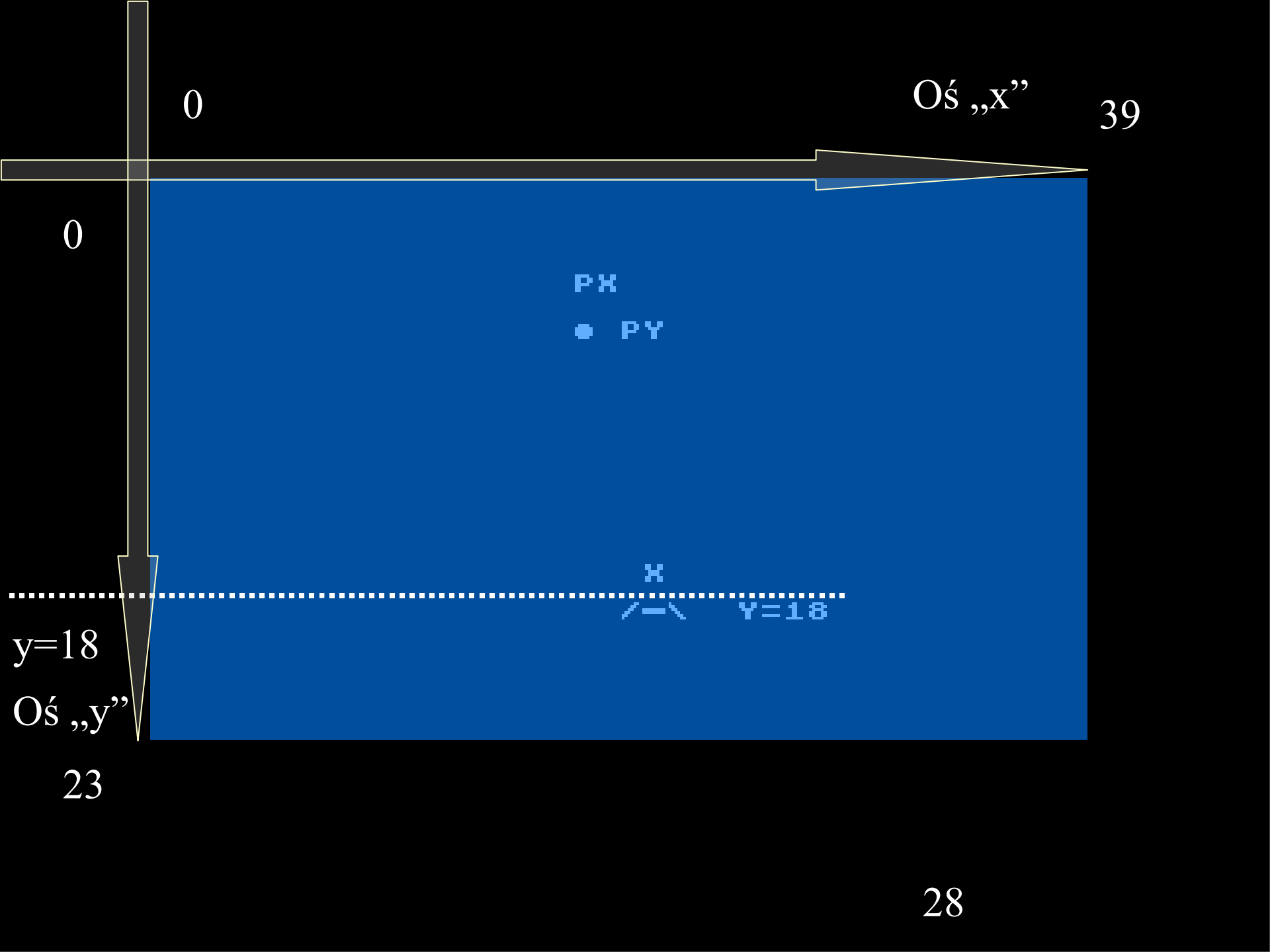

0
Oś „x”
39
0
PX
PY
Oś „y”
23

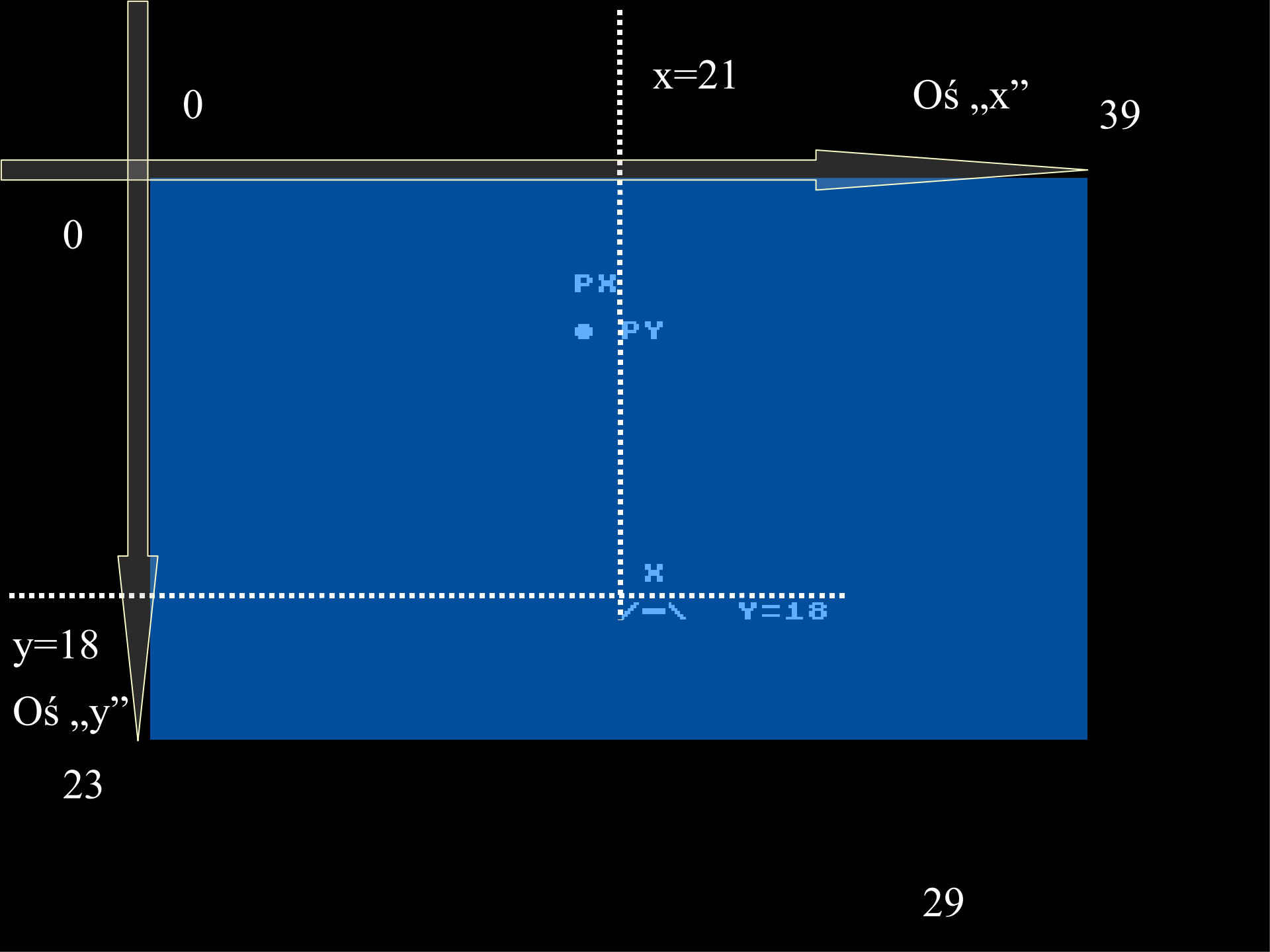

0
x=18
Oś „x”
39
0
PX
PY
Oś „y”
23
25




```
130 POSITION PX,PY:? "●"
```

Drugi przykład


0
Oś „x”
39
0
PX
PY
X
Y=18
y=18
Oś „y”
23


0
x=21
Oś „x”
39
0
PX
PY
X
Y=18
y=18
Oś „y”
23

Kasowanie zawartości ekranu

GRAPHICS 0 (Skrót GR.0) – uruchamia jeden z 16 trybów graficznych Atari (w tym wypadku tryb tekstowy)

lub

To jest znak specjalny. Osiąga się go:

1. Naciśnij klawisz Esc

2. Naciśnij Shift+Home

W ten sposób można tworzyć inne specjalne znaki np. strzałki, tabulatory, insert, delete itp.

Kolory

Po wykonaniu polecenia np.

GRAPHICS 0

Warto ustawić nowe wartości dla rejestrów kolorów

Rejestry:

709 – odpowiada za jasność tekstu

710 – odpowiada za kolor tła tekstu

712 – odpowiada za kolor ramki dookoła ekranu (zwykle =0)

Przykład białe napisy na czarnym tle (czyli pasuje klimatem do naszej gry;)

POKE 709,14: POKE 710,0

gdzie:

709 – odpowiada za jasność tekstu

710 – odpowiada za kolor tła tekstu

(teraz 710 i 712 wynoszą 0 – więc ramka jest niewidoczna)

Wyznaczanie koloru dla rejestrów 710 i 712:

Kolor*16 + Odcień

Gdzie:

Kolor – wartość od 0 do 15

Odcień – wartość od 0 do 15 (wartości nieparzyste są pomijane, zero jest parzyste ;)

Np. POKE 710,0 – oznacza kolor czarny (Kolor=0) w odcieniu 0 – czyli najciemniejszym dostępnym.

INPUT

INPUT zmienna – wprowadzenie danej przez użytkownika

np.

10 ? ”Podaj wiek”

20 INPUT wiek

30 ? ”Twoj wiek wynosi ”; wiek; ” lat.”

Przykład wykorzystania INPUT:

```
READY
10 ? "PODAJ WIEK
20 INPUT WIEK
30 ? "TWOJ WIEK WYNOSI "; WIEK; " LAT.
"

RUN
PODAJ WIEK
?5
TWOJ WIEK WYNOSI 5 LAT.

READY
```

Wprowadzenie wartości typu tekstowego:

```
READY
10 DIM A$(100)
20 ? "WPROWADZ IMIE"
30 INPUT A$
40 ? "WPROWADZILES IMIE "; A$
RUN
WPROWADZ IMIE
?ADAM
WPROWADZILES IMIE ADAM

READY
```

Sprawdzanie co zostało wprowadzone do tablicy tekstowej A$:

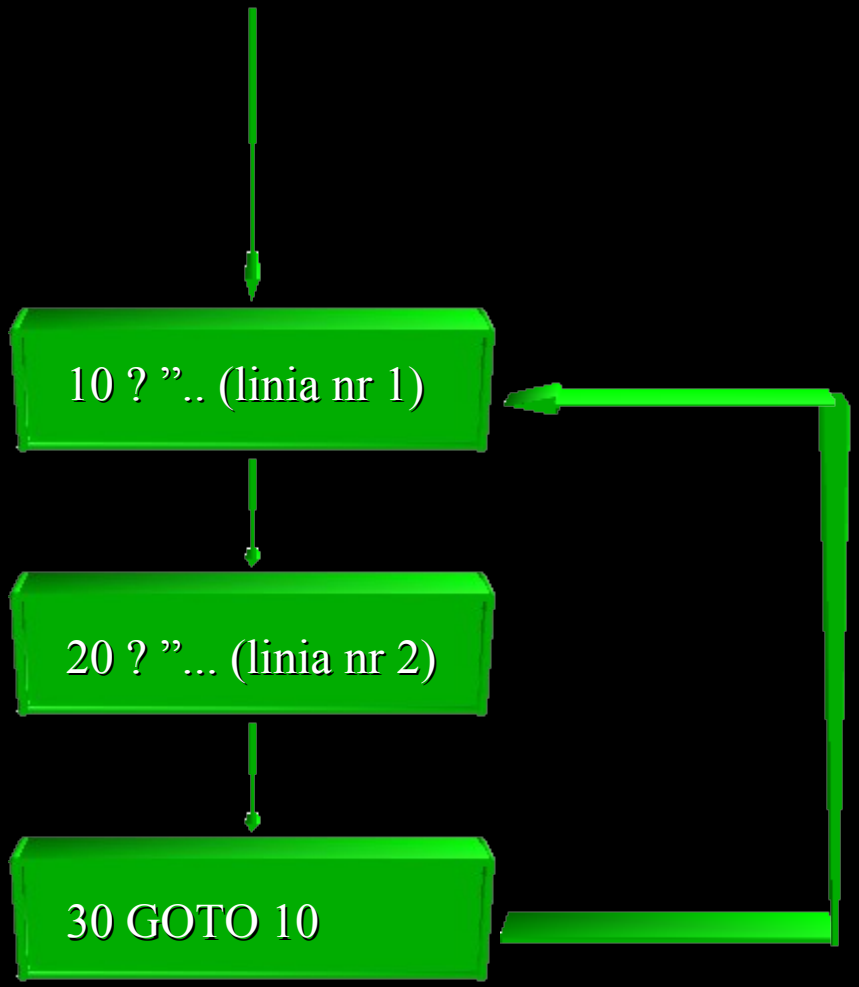
```
40 IF A$=”ADAM” THEN ? „Hasło zgadza się”: GOTO x/
END

50 ? „Udzieliłeś niepoprawnej odpowiedzi

60 END
```

Pętla GOTO

Pętla GOTO:



Zwykła pętla to polecenie

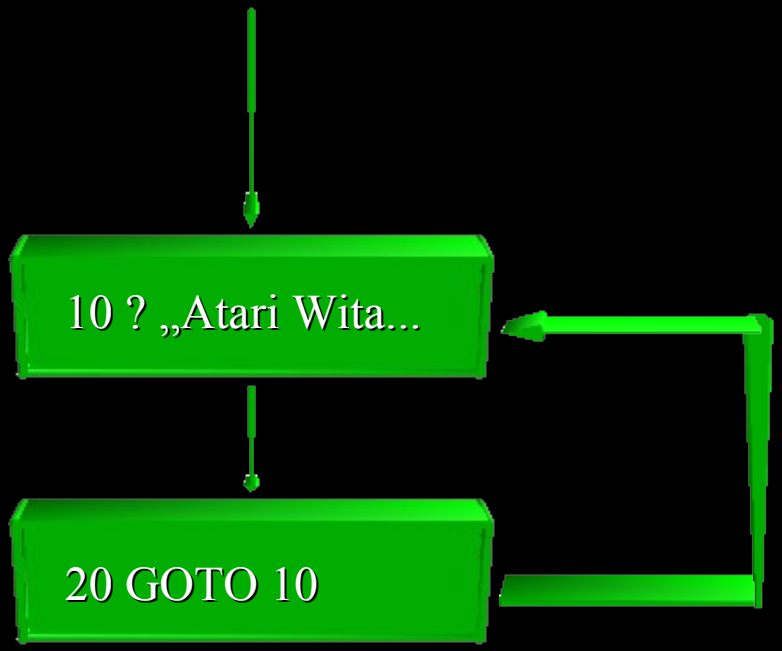
GOTO numer_linii_programu

np.

10 ? „Atari Was wita!”

20 GOTO 10

RUN



lub

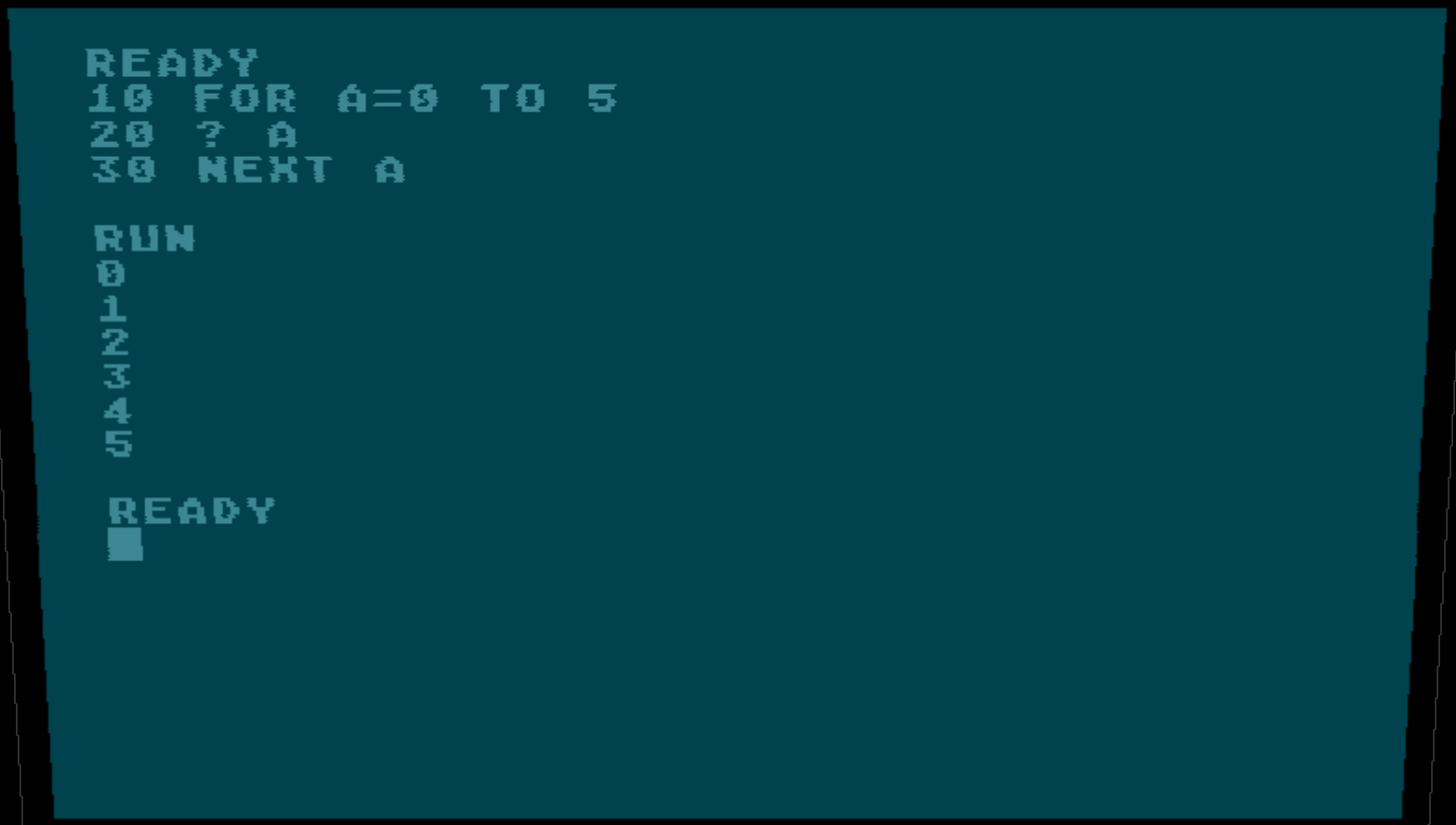
```
READY
10 ? A:A=A+1:GOTO 10
RUN
0
1
2
3
4
5
6
7
8
9

STOPPED  AT LINE 10
```

Pętla FOR

Pętla FOR:

Odlicza we wskazanym zakresie, za pomocą zmiennej (sterującej pętlą), np.



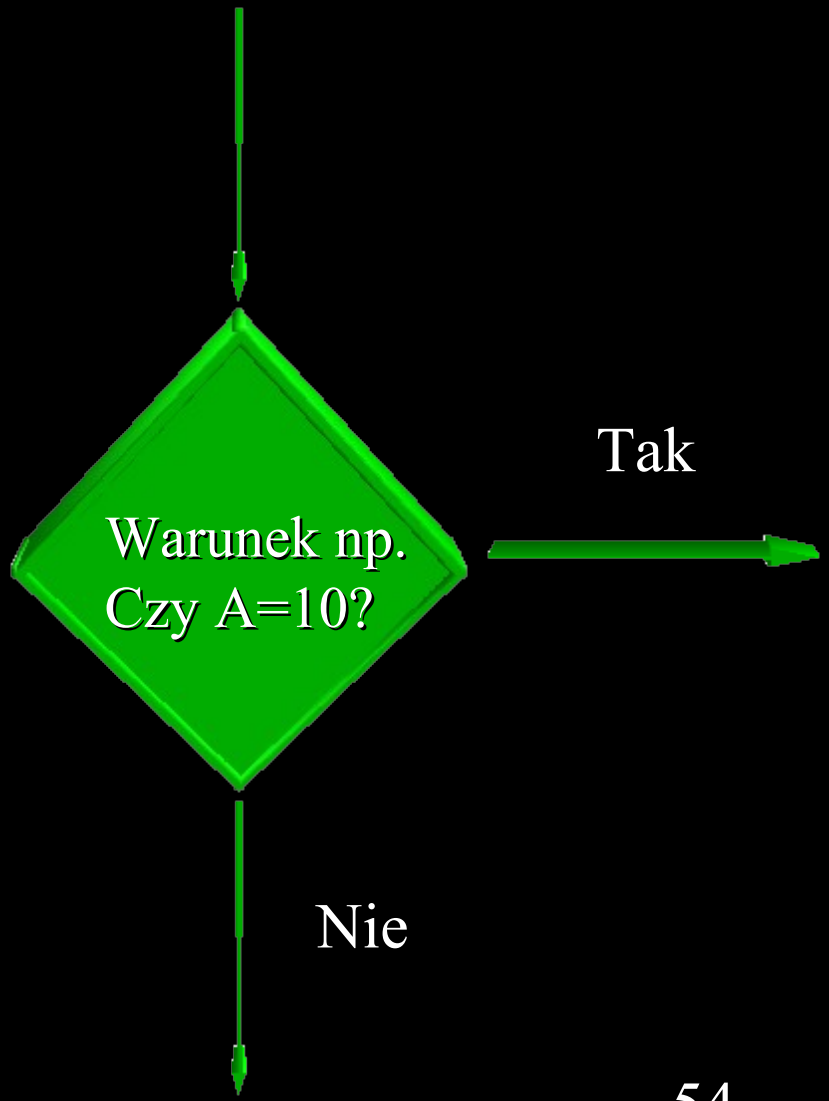
Można też nadać skok np.

```
FOR A=0 TO 10 STEP 2: ? A: N.A
```

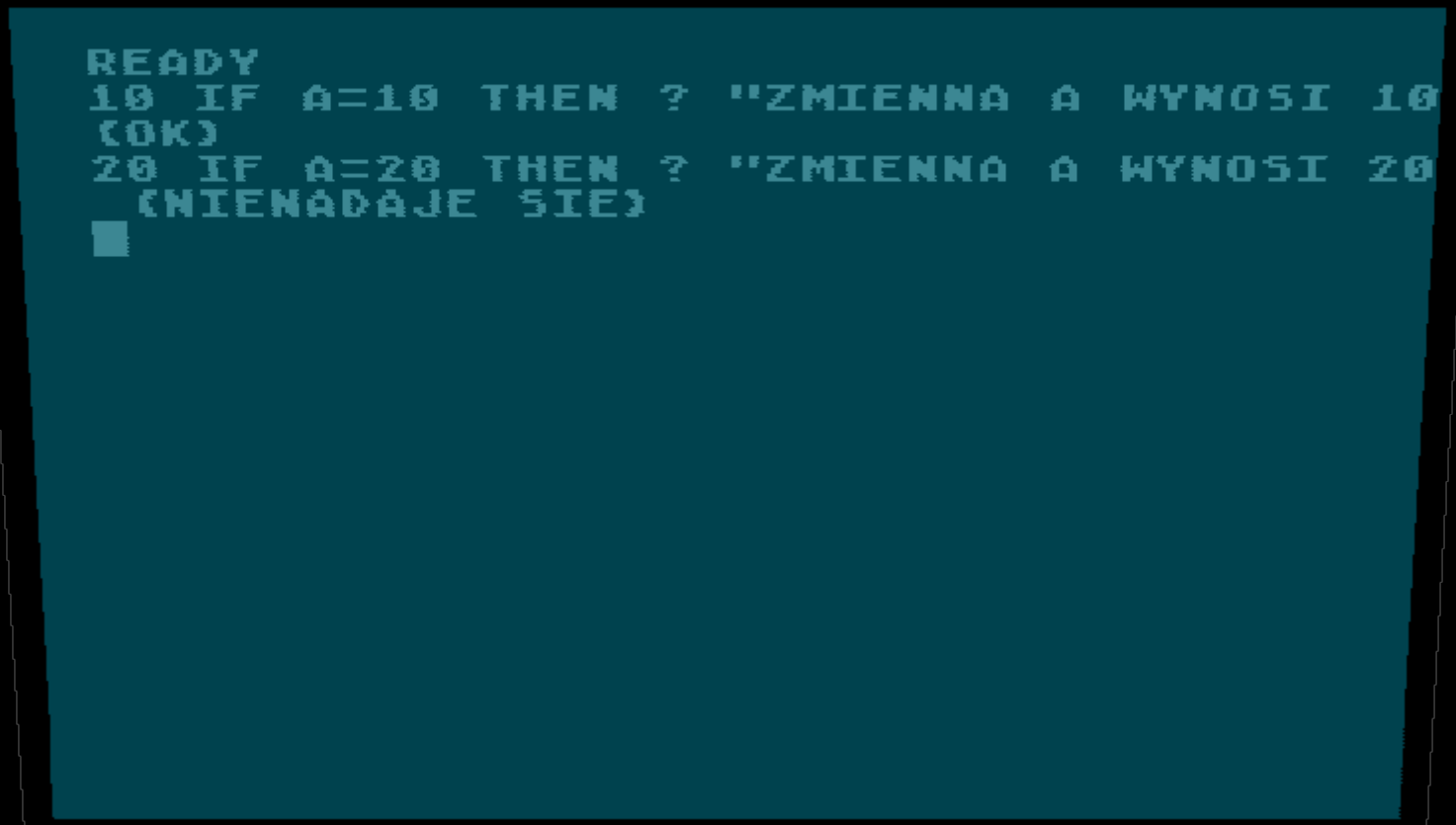
gdzie N. to skrót od NEXT

Warunek IF

Warunek IF:



Warunek IF i THEN



W Atari BASIC nie istnieje ENDIF oraz ELSE. Rolę ENDIF pełni koniec linii, dlatego w dłuższym kodzie, warto skorzystać z GOTO lub GOSUB.

Warto łączyć IF i GOTO (jeszcze lepiej GOSUB), np.

```
100 IF A=10 THEN G. 1000
```

Domyślna odpowiedź (default)

Często potrzebujemy domyślnej odpowiedzi np. na działania użytkownika, np.

```
100 ? "PODAJ ROZMIAR (5 LUB 10)
200 INPUT ROZM
210 IF ROZM=5 THEN GOTO 400
220 IF ROZM=10 THEN GOTO 500
290 ? "PODALES NIEPRAWIDLOWA WARTOSC": G. 100
400 ...
499 End
500 ...
```

Domyślna odpowiedź (linia numer 290):

```
READY
100 ? "PODAJ ROZMIAR (5 LUB 10)
200 INPUT ROZM
210 IF ROZM=5 THEN GOTO 400
220 IF ROZM=10 THEN GOTO 500
290 ? "PODALES NIEPRAWIDLOWA WARTOSC":
 G. 100
```

Podprogram

Podprogram:

GOSUB numer linii – skok do podprogramu (skrót GOS.)

RETURN – powrót z podprogramu

Np.

```
10 GOSUB 100
90 END
100 ? „To jest podprogram, który często się przydaje w grze
199 RETURN
```

Podprogram GOSUB i RETURN:

Podprogram musi być uniwersalnie napisany, tak aby z każdego miejsca kodu gry można było wykonywać do niego skok typu GOSUB.

Wtedy ten sam kod obsłuży dane zdarzenie w różnych miejscach rozgrywki. Np. handel, rozmowa, poszukiwanie sklepu/kontaktu może odbywać się identycznie (po skoku to podprogramu) bez względu na to, w którym miejscu rozgrywki się znajdujemy.

Inne przykłady: kod celowania podczas strzelania (np. z wykorzystaniem: joysticka, pistoletu świetlnego, myszki, paddle itp.), kod sprawdzający czy trafiliśmy, dokonanie zakupu, wyświetlenie danych na ekranie, boss itp.

Wybór opcji

Np. gracz decyduje co robi

Gracz podejmuje decyzje co robi:

10 ? „Witaj w grze pt. PK”: ?”Znajdujesz się na ulicy, co chcesz teraz zrobić?

100 ? „Możesz wykonać: 1 – idziesz dalej, 2 – handlujesz, 3 – szukasz lokalnego kontaktu, 4 - pytasz o..., 5 – kończysz grę”

110 INPUT co_robi

120 IF co_robi=1 THEN GOTO 200: REM w linii 200 znajduje się podobna treść co w 100-190, ale dotyczy innej lokalizacji

130 IF co_robi=2 THEN GOSUB 1000: REM tu jest podprogram do handlu

140 IF co_robi=3 THEN GOSUB 2000: REM tu jest podprogram do obsługi poszukiwania kontaktu (lub porażki)

150 IF co_robi=4 THEN GOSUB 3000: REM tu znajduje się podprogram obsługujący konwersacje

160 IF co_robi=5 THEN GOTO 10: REM poddałeś się, gra się skończyła

180 ? „Wybrałeś nieistniejącą opcję

190 GOTO 100

Baza danych

Dane zapisujemy w liniach DATA (gdzie numery linii nie mają decydującego znaczenia) np.

10000 REM - DOSTEPNE ROZMIARY -

10001 DATA 4,5,6,7,8,11,12,16,19,20

Kolejne rozmiary możemy dodawać w tej samej linii lub w kolejnych. Zwiększanie numerów linii o zadany krok nie ma znaczenia dla działania programu, za każdym razem odczytywana będzie kolejna dostępna wartość, np.

10000 DATA 4

10010 DATA 5

10020 DATA 6

10030 ...

READ zmienna – odczyt kolejnej (lub pierwszej) danej z linii DATA, np.

```
READY
L.

10 READ DANA
20 ? "PIERWSZA WARTOSC W BAZIE DANYCH
WYNOSI: ";DANA
99 END
10000 REM - DOSTEPNE ROZMIARY -
10001 DATA 4,5,6,7,8,11,12,16,19,20

READY
RUN
PIERWSZA WARTOSC W BAZIE DANYCH WYNOSI
: 4

READY
```

Kolejne wartości odczytujemy wykonując kolejne polecenie READ.

RESTORE – odczyt danych od początku (z linii DATA)

RESTORE numer_linii – odczyt danych od zadanej linii

np. każdy element gry rozpoczyna się od linii o przewidywalnym numerze np. 10000, 11000, 12000, 13000 itp. Mogą to być opisy lokalizacji, obiektów, postaci w grze (np. płatnych morderców mafii itp.)

np.

10000 DATA ”Kilerek”, 1000000: REM Pseudonim i cena

11000 DATA ”Kaprawy”, 600000

12000 DATA ”As”, 1500000

13000 DATA ...

Odczyt danych różnego typu do gry np. parametrów killera.

Killer ma różnego typu dane, które go opisują, np. imię czy pseudonim jest daną typu tekstowego.

Natomiast cena zlecenia jest zwykłą daną numeryczną (zwykła zmienna).

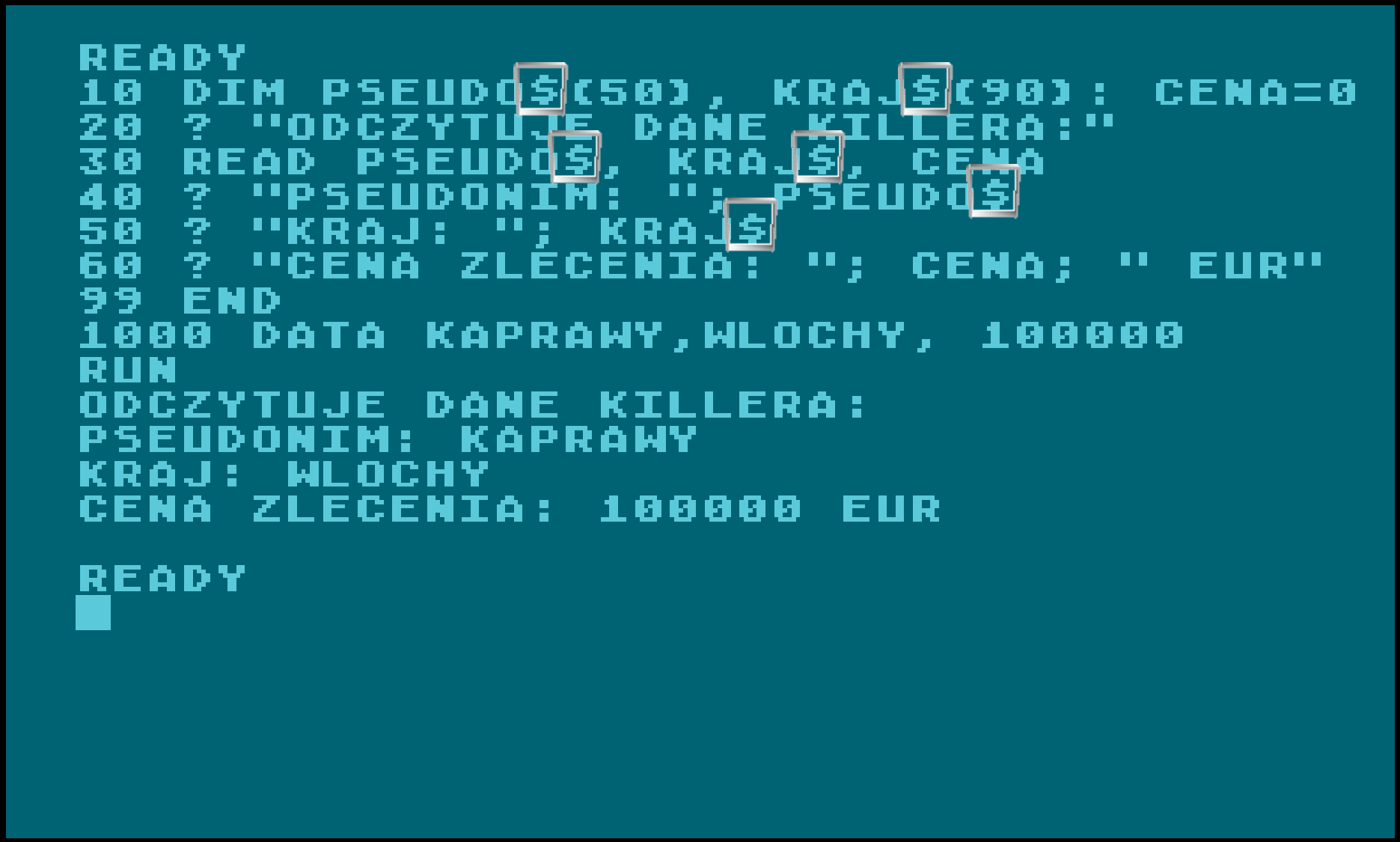
Skoro język Atari BASIC rozróżnia zmienne tych dwóch typów, musimy pamiętać, aby przy pobieraniu danych tekstowych nie zapomnieć o „$” przy nazwie zmiennej tekstowej.

Odczyt danych różnego typu do gry np. parametrów killera:

```
READY
10 DIM PSEUDO$(50), KRAJ$(90): CENA=0
20 ? "ODCZYTUJE DANE KILLERA:"
30 READ PSEUDO$, KRAJ$, CENA
40 ? "PSEUDONIM: "; PSEUDO$
50 ? "KRAJ: "; KRAJ$
60 ? "CENA ZLECENIA: "; CENA; " EUR"
99 END
1000 DATA KAPRAWY,WLOCHY, 100000
RUN
ODCZYTUJE DANE KILLERA:
PSEUDONIM: KAPRAWY
KRAJ: WLOCHY
CENA ZLECENIA: 100000 EUR

READY
```

W kolejnych liniach można wprowadzać kolejnych killerów.
np. w linii numer 1010 DATA Kilerek ...


READY
10 DIM PSEUDO$(50), KRAJ$(90): CENA=0
20 ? "ODCZYTUJE DANE KILLERA:"
30 READ PSEUDO$, KRAJ$, CENA
40 ? "PSEUDONIM: "; PSEUDO$
50 ? "KRAJ: "; KRAJ$
60 ? "CENA ZLECENIA: "; CENA; " EUR"
99 END
1000 DATA KAPRAWY,WLOCHY, 100000
RUN
ODCZYTUJE DANE KILLERA:
PSEUDONIM: KAPRAWY
KRAJ: WLOCHY
CENA ZLECENIA: 100000 EUR
READY

It's a trap!

Podczas korzystania z READ/DATA może dojść do jakiegoś błędu np. mogą skończyć się dane itp.

Do obsługi tego typu zdarzeń służy polecenie

TRAP numer_linii

Obsługa takiego błędu może wyglądać w ten sposób:

0 ? „Witaj w grze...”: TRAP 32000

32000 ? „Wystąpił nieoczekiwany błąd w grze”: RESTORE

0 ... : TRAP 32000

32000 ? „Wystąpił nieoczekiwany błąd w grze”: RESTORE

Podczas zaistnienia dowolnego błędu zostanie wykonana linia o numerze 32000

Wykona ona RESTORE czyli dane będą czytane od początku (czyli od pierwszej linii zawierającej dane w DATA). W wielu przypadkach to wystarczy, choć może być tak, że program czyta dane tekstowe, a na początku są dane numeryczne, więc program nadal nie będzie działał prawidłowo (warto wtedy dodać RESTRORE numer_linii, tak aby się wszystko zgadzało).

Dodatkowo można napisać, zamiast:

32000 ? „Wystąpił nieoczekiwany błąd w grze”: RESTORE

np.

32000 ? „Wystąpił nieoczekiwany błąd w grze”: RESTORE: GOTO 10

Dzięki temu po zaistnieniu dowolnego błędu, nie tylko zostanie wykonane RESTORE, ale zostanie wykonany skok do początku programu. Co prawda gra się zakończy (gracz utraci rozgrywkę i zdobyte zasoby), lecz jest to bezpieczne i gra będzie dalej działać prawidłowo.

Założenie: od linii 10 rozpoczyna się kod gry np. czołówka

0 TRAP 32000

5 GRAPHICS 0

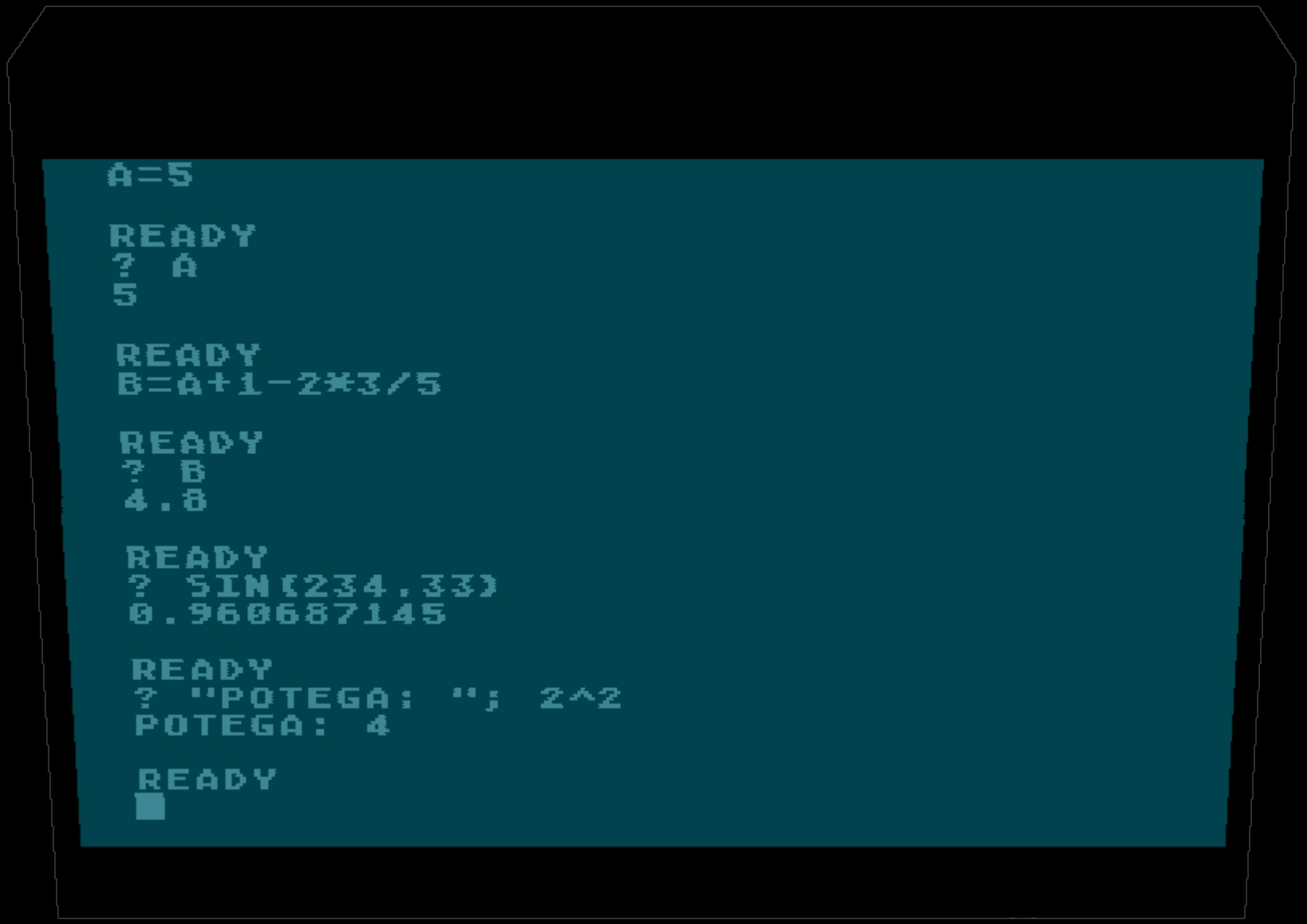
10 ? „Witaj w grze pt. PK, ...

32000 ? „Wystąpił nieoczekiwany błąd w grze”: RESTORE: GOTO 10

Warto się upewnić, aby w linii 32000 nie wykonywać skoku GOTO do linii, w której kasowany jest obraz (czyli tu w linii 5), gdyż wtedy nie zdążymy przeczytać komunikatu z linii 32000.

Operacje arytmetyczne

Operacje arytmetyczne:



Zwiększanie i zmniejszanie wartości zmiennej np.

SCORE=SCORE+100

Zwiększa zmienną SCORE o 100.

ŻYCIA=ŻYCIA-1

Zmniejsza zmienną ŻYCIA o 1.

Inne

Kilka operacji w jednej linii oddzielamy za pomocą „:”

np.

10 INPUT A: ? A: END

Uwaga: maksymalna długość linii kodu jest ograniczona do 3 wierszy widocznych na ekranie.

PRINT (skrót ?)

Warto dodać odstęp kilku linii, aby oddzielić różne teksty od siebie, zwiększa to czytelność gry.

Aby zrobić 2 linie odstępu piszemy:

200 ? : ?

3 linie:

200 ? : ? : ?

itp.

Średnik pozwala łączyć wyświetlanie w komendzie PRINT (skrót PR. lub ?)

np.

? ”Ala ma ”; ilosc_kotow; ” kotow.”

> Ala ma 5 kotow.

Dla ilosc_kotow=5

Kolejne komendy z średnikiem mogą znajdować się w kolejnych liniach programu. Np.

10 ? ”Ala ma ”;

20 ? ilosc_kotow; ” kotow.”

Usuwanie całej linii kodu.

Jeśli dana linia zawiera niepotrzebną treść możemy:

1. Zastąpić linię

Wystarczy wpisać linię od nowa z numerem już istniejącej linii. Linia zostanie zastąpiona

2. Skasować

Wystarczy wpisać numer linii + klawisz RETURN (Enter na emulatorze)

REM – komentarz w kodzie programu (skrót RE.)

POKE adres,wartość – ustaw wartość komórki pamięci

POKE 710,0

PEEK(adres) – odczytaj wartość komórki pamięci

? PEEK(710) lub Kolor=PEEK(710)

RND(numer generatora) – sprzętowy generator wartości losowych, np.

? INT(RND(0)*10)

> 2

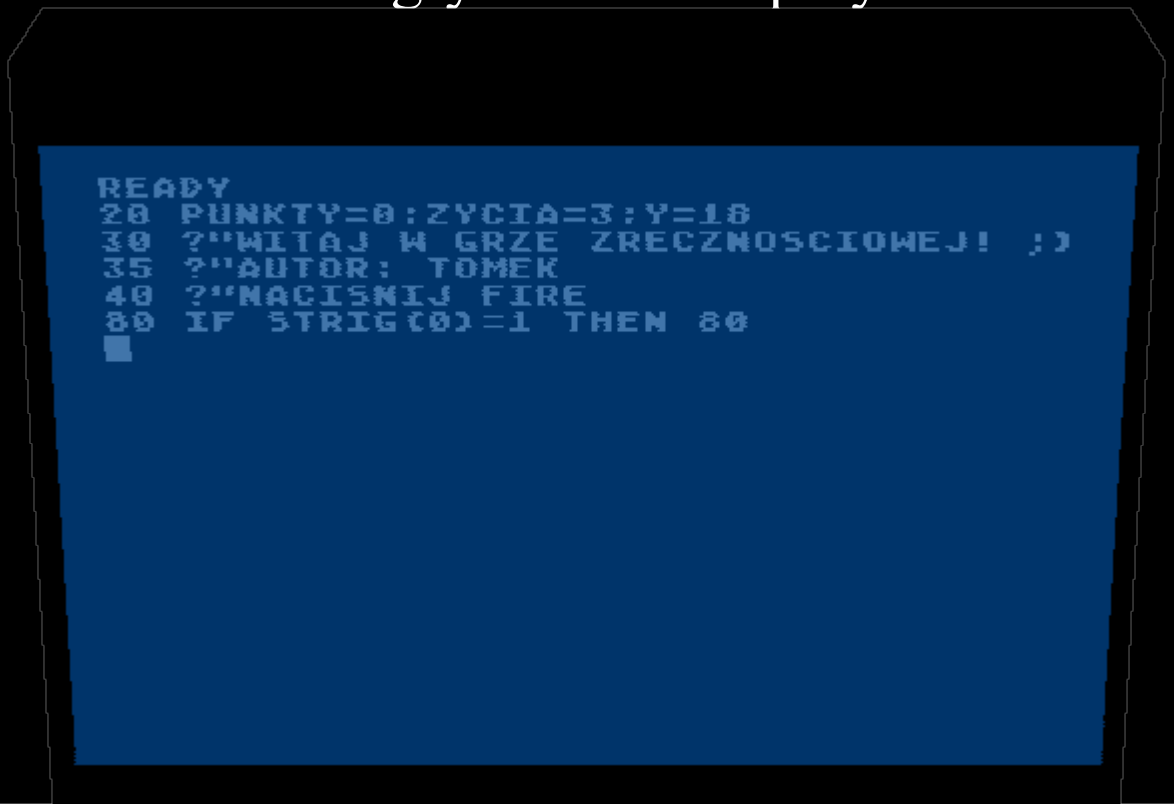
INT(wartość) – zaokrąglenie wartości do liczb całkowitych

STICK(numer joysticka) – odczyt aktualnego kierunku joysticka (4 bity), np.

```
10 ? STICK(0): G. 10
```

STRIG(numer joysticka) – odczyt przycisku fire (0 lub 1)

Przykładowa czołówka gry sterowana przyciskiem fire:



Joystick musi być skonfiguro wany w emulatorze.

Inne komendy Atari BASIC:

COLOR, LET, TRAP, BYE, COM, CLOSE, CLR, OPEN, STATUS, NOTE, POINT, XIO, ON, STOP, POP, GET, PUT, GRAPHICS, PLOT, DRAWTO, DOS, SETCOLOR, LOCATE, SOUND, LPRINT, CSAVE, CLOAD, LET (domyślne), PADDLE, PTRIG.

DEG, RAD.

Przykład dźwięku w grze:

Wybuch:

FOR Q=15 TO 0 STEP -0.5: SOUND 0,150,8,Q:N.Q

gdzie:

SOUND numer_kanału_dźwięku (od 0 do 3), częstotliwość (od 0 do 255), obwiednia (od 0 do 15 tylko parzyste wartości), głośność dźwięku (od 0 do 15)

Przykład dźwięku w grze:

Wybuch:

np. w grze podczas wystąpienia kolizji pocisku z pojazdem gracza (zmienne Pocisk X i Pocisk Y oraz X):

```
READY
180 IF PY=18 AND PX=X THEN ZYCIA=ZYCIA
-1:PY=22:FOR Q=15 TO 0 STEP -0.5:SOUND
 0,150,8,Q:N.Q
```

Inne słowa kluczowe:

NOT, OR, AND

STR$, CHR$

USR

ASC, VAL, LEN, ADR, ATN

COS, SIN, FRE, EXP, LOG, CLOG, SQR, SGN, ABS

Błędy w Atari BASIC (Error)

Podczas edycji programu błędy są od razu wskazywane.

```
50 ERROR-    ?:S SAS''SDA
```

Błędy podczas wykonywania programu są zwracane jako

ERROR- nr AT LINE n

```
READY
RUN
?TOMEK
-

ERROR-    17 AT LINE 50
```

```
READY
RUN
?TOMEK
_

ERROR-   17 AT LINE 50
```

Kody błędów można sprawdzić pod tym adresem (kody od 2 do 21):

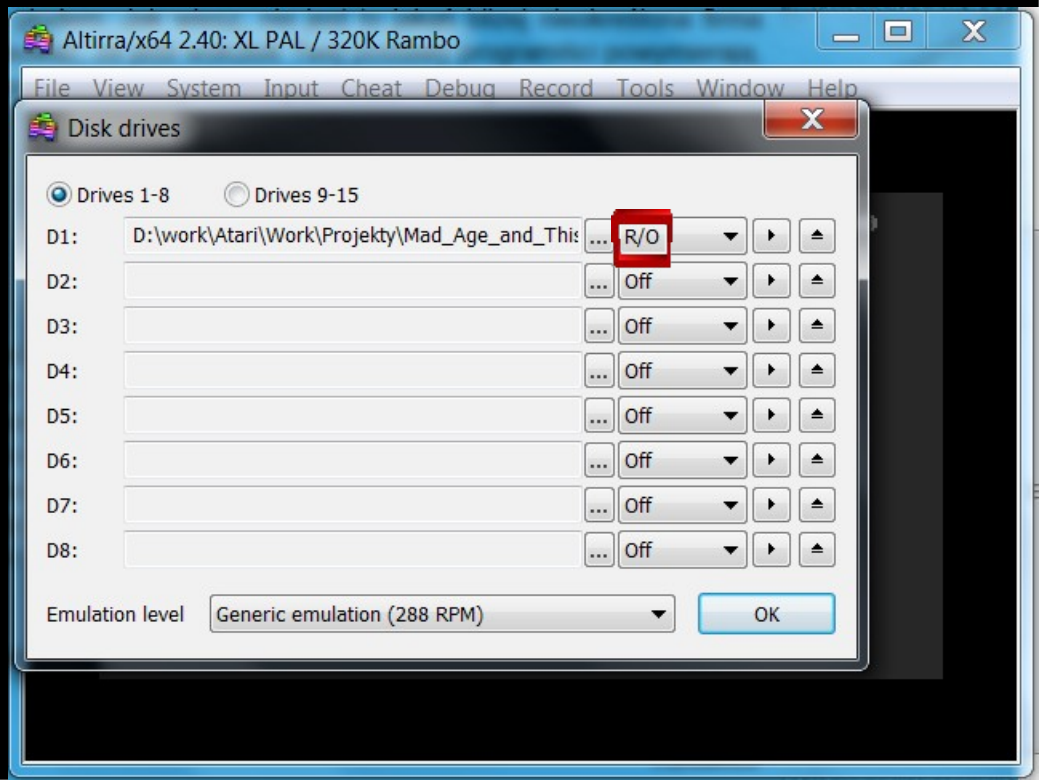
http://atariki.krap.pl/index.php/Kody_b%C5%82%C4%99d%

Błędy wejścia/wyjścia (np. operacje na plikach, czyli błędy zgłaszane przez DOS) mają wyższe numery od 21.

Zwykle zdarza się, że w emulatorze plik obrazu dysku ATR jest zabezpieczony przed zapisem.

Menu: File > Disk Drives...

lub Alt+Shift+D



Błędy interpretera

Interpreter Atari BASIC ma błędy w implementacji, wiele z tych błędów znajduje się też w innych popularnych językach BASIC dostępnych na Atari.

Opis błędów znajduje się w rozdziale „Błędy w Atari BASIC”:

http://atariki.krap.pl/index.php/Atari_BASIC

! To jest tylko dla ambitnych! ;)

Korzystanie z podstawowych 10 rozkazów Atari BASIC (slajd 18) nigdy nie spowoduje napotkania takiego błędu.

Edytor systemowy Atari

Edytor systemowy Atari umożliwia edytowanie całoekranowe kodu programu.

Wpisanie komendy BASIC i zatwierdzenie RETURN (Enter na emulatorze), sprawdza składnie wyrażenia i wykonuje je. np.

? „Witaj!

Polecenie poprzedzone numerem linii – zapamiętuje je w pamięci. (jest sprawdzana poprawność składni).

Dopuszczalna numeracja linii od 0 do 32767.

Zaleca się odstęp między kolejnymi numerami linii np. 10 lub 100 (dla interpretera nie ma to znaczenia). Dzięki temu możliwe jest wstawienie kodu między wcześniej wpisany.

Poruszanie się strzałkami po całym ekranie

Na Atari poruszamy kursorem za pomocą strzałek znajdujących się po prawej stronie klawiatury (bardzo praktyczne rozwiązanie, które umożliwia znacznie szybsze pisanie niż na klawiaturach pecetowych).

Klawisze strzałek działają wyłącznie z klawiszem Control.

Emulatory zwykle działają w trybie domyślnego przyciśniętego klawisza Control, więc klawisze strzałek zawsze poprawnie działają.

W przeciwnym wypadku należy w emulatorze zmienić właściwą opcję. W emulatorze Altirra:

Np. Input > Keyboard (Options) – Arrow key mode

Kasowanie zawartości ekranu (np. gdy jest zapisany kodem programu):

Klawisze:

Shift + Clear (Atari)

Control + Home (Emulator)

Alternatywnie można wykonać polecenie GR.0 lub nacisnąć klawisz Reset (F5 na emulatorze).

Zyskanie wolnej linii (gdy ekran jest zapisany kodem)

1. Klawisz Break

Naciśnięcie klawisza Break (Pause na emulatorze) przechodzi do kolejnej linii. Naciśnięcie kilka razy da nam nowe miejsce do programowania.

2. Rozsunięcie tekstu

Shift + Insert (Atari i na emulatorze)

(zyskujemy jedną linię, gdzie możemy pisać, należy uważać aby nie rozdzielać długiej linii kodu w jej wnętrzu bo edytor może zinterpretować nową zawartość jako dalszy ciąg poprzedniej itp.)

Rozsuwanie zawartości linii w lewo i prawo w miejscu kursora:

Klawisz Insert (Atari i Emulator) przesuwa zawartość linii za kursorem, dając miejsce na nową treść.

Klawisz Delete (Atari i Emulator) przesuwa zawartość linii za kursorem, w lewo. To co się znajduje za kursorem jest za każdym razem bezpowrotnie kasowane.

UWAGA!

Przypadkowe uszkodzenie linii kodu **nie wymaga wprowadzenia poprawek** (szczególnie, gdy coś całkowicie zniknęło).

Linia w pamięci znajduje się cały czas zapamiętana poprawnie.

Dlatego nic nie należy robić!

Wystarczy wpisać (dla linii numer 100):

L.100

Linia zostanie wyświetlona w takiej formie w jakiej była wcześniej zapamiętana (czyli nieuszkodzonej).

Błędem jest zatwierdzenie uszkodzonej linii klawiszem Return (Enter w emulatorze).

W takim wypadku zostanie zapamiętana uszkodzona treść linii...

Dlatego odradzam automatyczne naciskanie (wielokrotnie) klawisza Return !

Każde naciśnięcie Return sprawdza i zapamiętuje/wykonuje to co znajduje się (przypadkiem) w danej linii.

np. jeśli przypadkiem coś zaczyna się od numeru to zostanie skasowana linia o tym numerze (!!) i zastąpiona tą przypadkową zawartością na ekranie...

Trik

Jeśli potrzebujemy wpisać kilka podobnych linii kodu – nie trzeba ich pisać kilka razy.

Np.

100 IF A=1 THEN G.500

Pisząc podobną kolejną linię cofamy kursor do góry i zmieniamy w niej tylko to co chcemy, czyli numer linii na kolejny i w warunku A=2, zatwierdzamy Return (Enter na emulatorze) i gotowe;)

110 IF A=2 THEN G.777

Polecenie LIST (skrót L.) wyświetli nam kod programu – możemy sprawdzić, czy wprowadziliśmy wszystko poprawnie.

Szybkie przesuwanie kursora:

Klawisz Tab w Atari ma inne znaczenie: przesuwa kursor o 9 znaków (lub 6 z marginesem).

Umożliwia to szybkie poruszanie się po zawartości linii (szczególnie w długich liniach się to przydaje).

Jest to nieco podobne w użyciu do kombinacji Control+strzałki (lewo lub prawo) na obecnych edytorach pecetowych.

Znaki w inwersji:

Zmiana trybu następuje po naciśnięciu:

1. Specjalnego klawisza na klawiaturze Atari (prawy dolny róg, symbol czarnego i białego).

2. Klawisz End na emulatorze

Znaki tego typu są wykorzystywane przez sprzęt Atari w innych trybach tekstowych. W trybie 0 (GRAPHICS 0), służy jedynie do zamiany kolorów liter i tła (też ma istotne znaczenie w trikach).

Zadania dla grup

1. Każda grupa ma za zadanie dodać do gry 1 lokalizację

Nadać jej nazwę (miasto/miejsce)

Dodać zdarzenia jakie mogą/muszą być obsłużone

Kod każdej lokalizacji musi się znaleźć w liniach g000 dla g – wynoszącego numer grupy np. grupa 1 to 1000 itp. 2 to 2000 itp.

Każda nazwa lokalizacji ma się znaleźć w linii DATA od numeru 100g0
dla g – wynoszącego numer grupy np. grupa 1 10010

2. Każda grupa ma dodać do gry jednego killera, któremu będzie można w grze zlecić zabójstwo.

Nazwa killera znajduje się w liniach od 20000 np. 20010 dla grupy 1.
(napisać L. 20000-29999 aby wyświetlić kod bazy danych)

2. Każda grupa ma dodać do gry jedną osobę będącą celem dla kilerów.

Nazwa celu dla killera znajduje się w liniach od 30000 np. 30010 dla grupy 1

W linii 30009 znajduje się przykład jak może wyglądać linia 30010, 30020, ...

Dodatkowo:

Grupa 1: Wykonuje intro gry
(linie 500-998)

Grupa 2: zdarzenie: wygrana
(linie 8000-8199)

Dodatkowo:

Grupa 3: zdarzenie: przegrana
(linie 9000-9199)

Grupa 4: podprogram wyświetlający cele
(linie 9800- 9899 <- tu macie programować)
na wzór kodu z 9900-9930)

Dodatkowo:

Grupa 5: czołówka gry
(linie 100-499)

Ściągamy pliki

Ściągamy pliki

http://tdc.pigwa.net/UJ/Podstawy_Atari
(http://tdc.pigwa.net/UJ/Podstawy_AtariBASIC_i_zadania.ppt)
lub
http://tdc.pigwa.net/UJ/Podstawy_Atari
(http://tdc.pigwa.net/UJ/Podstawy_AtariBASIC_i_zadania.pdf)

http://tdc.pigwa.net/UJ/Prg.zip

http://tdc.pigwa.net/UJ/Proj1.atr

Ściągamy pliki

katalog: http://tdc.pigwa.net/UJ/

UWAGA!

Każda grupa zmienia nazwę pliku Proj1.atr

na numer odpowiadający jej numerowi grupy (Proj2.atr, Proj3.atr, ...)

UWAGA!

To samo się tyczy pliku w ATR, czyli: PROJ1.BAS

Każda grupa zmienia nazwę tego pliku na numer odpowiadający jej numerowi grupy (Proj2.atr, Proj3.atr, ...).

UWAGA!

Jak zmienić nazwę pliku:
przechodzimy do DOSa, wykonując polecenie:
DOS

następnie (REN od REName)

REN PROJ1.BAS PROJg.BAS

Opis struktury kodu gry
(z pliku proj1.ATR
i pliku
"PROJ1.BAS")

(wczytujemy LOAD "D:PROJ1.BAS")

40 – 60 inicjowanie gry, deklaracje zmiennych, ustawienia kolorów itp.

100 - 180 czołówka gry
dopiszcie coś tutaj;)

(może jakiś sławny cytat – jako przesłanie gry?)

500 – 998 Intro gry
Niech wprowadzi gracza w klimat gry oraz przedstawi cel gry (ew. postacie w grze itp.)

Warto zacząć od przedstawienia gracza z imienia i nazwiska lub pseudonimu, jakie jest jego zawód, cel gry, co mu grozi itp. Jak może przegrać (niekoniecznie musi to wiedzieć)

1000 – 1999 – kod od 1 lokalizacji

2000 – 2999 – kod 2 lokalizacji

3000 – 3999 – kod 3 lokalizacji

...

6000 – 6999 – kod 6 lokalizacji

1000 – 1999, 2000 – 2999, 3000 – 3999, ... , 6000 – 6999

W tych zakresach należy umieszczać cały kod z treścią gry jaką zaprojektujecie (np. podprogramy do zdarzeń itp.).

1000 – 1999, 2000 – 2999, 3000 – 3999, ... , 6000 – 6999

Uruchamiamy nasz kod np. G.2000 (grupa 2)

Wyświetlamy do edycji:
L. 2000 lub L. 2000,2999
(dla grupy 2)
(L. to skrót od LIST)

Warto aby każdy zespół w swojej lokalizacji umieścił:
- sprawdzanie czy gracz wygrał (skok do 8000)
- dać mu możliwość poruszania się do następnej lokalizacji lub do wybranych (lub tylko cofnąć się, bo np. czegoś mu brakuje itp.)
- dać graczowi jakieś wyzwanie (może coś stracić, może zyskać)

Warto aby każdy zespół w swojej lokalizacji umieścił:
- dać możliwość zamówienia killera na jakiś cel (tematyka gry!) i zbadać czy to zlecenie się powiodło (np. losowo lub z uwzględnieniem reputacji danego killera)

Warto aby każdy zespół w swojej lokalizacji umieścił:

- można dodać jakieś zdarzenia losowe, które np. przenoszą w inne miejsce (nie za często), gracz traci życie, zyskuje, traci/zyskuje kasę itp.

W linii g010 znajduje się test czy nastąpił koniec gry.
Np.
1010 IF ZYCIA<1 THEN GOTO 9000

8000 – 8199 – Wygrana, ten kod się wykona jeśli gracz wygra grę, należy go poinformować jaki jest wspaniały;) i co przez to osiągnął (cel gry) itp.

Dobrze jest wyświetlić wszystkie istotne zmienne (stan gry).

Warto gdzieś umieścić skok do 8000, aby było możliwe wygranie gry;)

9000 – 9199 – Przegrana, jeśli graczowi skończyły się życia, skaczemy do tego miejsca.

Należy gracza poinformować, jak bardzo słabym bandziorem był;)
Co zawalił i kto na tym zyskał;)

Dobrze jest wyświetlić wszystkie istotne zmienne (stan gry).

9000 – 9199 – Przegrana

Przed skokiem do 9000 wskutek jakiegoś zdarzenia warto gracza poinformować co się stało (np. trafił go snajper, wybuchła bomba (dlaczego?), zemścił się ktoś (kto? Za co?).
Dopiero tekst w 9000 jest bardziej ogólny (wspólny) i wyjaśni ostatecznie graczowi co stracił, czego nie osiągnął

9900 – Podprogram pobierający i wyświetlający dane killera

Skok (GOSUB) do podprogramu należy poprzedzić RESTORE, które wskaże konkretnego killera np. RESTORE 20010+coś*10 („coś” może być losowane).
Podprogram wywołuje się GOSUB 9900 (skrót GOS. 9900)

10000 – Baza danych lokalizacji w grze (każdy zespół ma swoją)

Lokalizacja to nazwa (zmienna „MIASTO$”).
Może to być miasto lub inne istotne miejsce.

Niech nazwa buduje klimat gry.

20000 – Baza danych celów dla klerów np. wg wzoru:
20010 DATA NICK,40000,8,
TU MA BYC OPIS KILLERA

NICK to nazwa, imię, przezwisko, nazwisko
40000 – to cena za którą składamy zamówienie

20010 DATA NICK,40000,8,
TU MA BYC OPIS KILLERA

8 – to reputacja (prawdopodobieństwo powodzenia akcji, że nie odmówi itp.)

„TU MA BYC OPIS KILLERA” - tu należy wprowadzić tekst opisujący go. Długość ograniczona do 200 znaków.

30000 – Baza danych celów
np.
30010 DATA NICK/NAZWISKO ,MNOZNIK_CENY,TRUDNOSC,TU MA BYC OPIS CELU

NICK/NAZWISKO – nick lub nazwisko celu

30010 DATA NICK/NAZWISKO ,MNOZNIK_CENY,TRUDNOSC,TU MA BYC OPIS CELU

MNOŻNIK_CENY – wartość przez jaką mnożymy cenę danego killera

TRUDNOSC – prawdopodobieństwo dorwania celu (np. w przedziale 0-10)

30010 DATA NICK/NAZWISKO ,MNOZNIK_CENY,TRUDNOSC,TU MA BYC OPIS CELU

TU MA BYC OPIS CELU – tekst opisujący cel (jeśli ma być dłuższy należy go rozbić na kilka linii DATA i tyle razy zrobić odczyt READ DANA1$ i ? DANA1$

Podstawowe zmienne (jakie zaproponowałem):

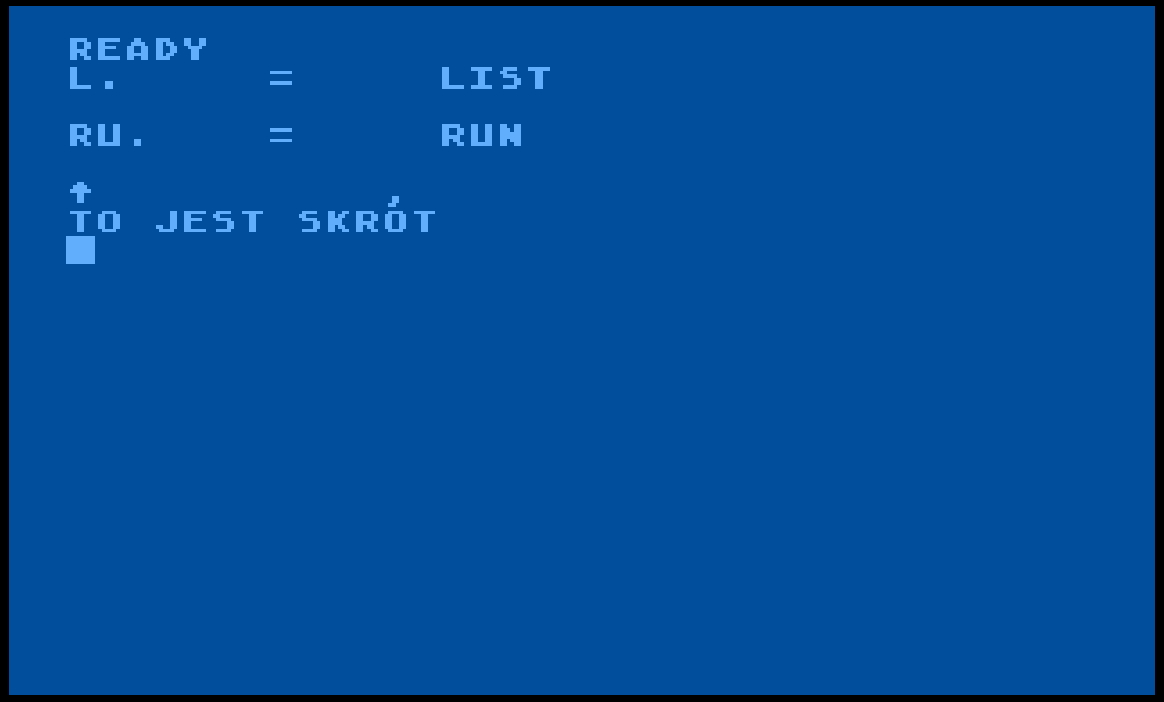
```
40 DIM IMIE$(100), ODP$(200),
PSEUDO$(50), MIASTO$(40),
DANA1$(200)
```

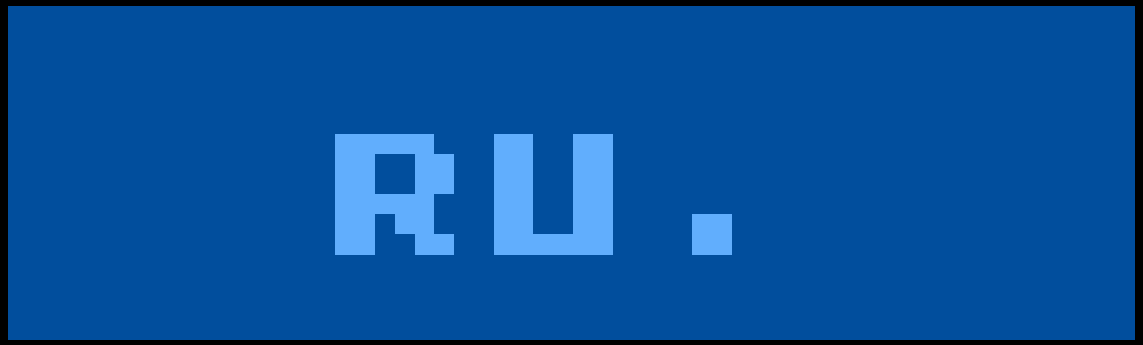
Podstawowe zmienne (jakie zaproponowałem):

60 ZYCIA=1: PUNKTY=0: KASA=0: REPUTACJA=1: REPUTACJAKILL=100: WYKZLEC=0

WYKZLEC – ilość WYKonanych ZLECeń

Uruchamiamy grę





Dziękuję za uwagę ;)