podstawy i języki programowania

pjp

Podstawy programowania

Część druga

Pierwsze programy

Autor
Roman Simiński

Kontakt
`roman.siminski@us.edu.pl`
`www.us.edu.pl/~siminski`

Niniejsze opracowanie zawiera skrót treści wykładu, lektura tych materiałów nie zastąpi uważnego w nim uczestnictwa.
Opracowanie to jest chronione prawem autorskim. Wykorzystywanie jakiegokolwiek fragmentu w celach innych niż nauka własna jest nielegalne.
Dystrybuowanie tego opracowania lub jakiejkolwiek jego części oraz wykorzystywanie zarobkowe bez zgody autora jest zabronione.

Problem do rozwiązania

Problem

Należy napisać program pozwalający na przeliczenie odległości podanej w kilometrach na mile amerykańskie.

Scenariusz działania programu:

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
Podaj odleglosc w kilometrach: 100
To w milach: 62.50
Nacisnij Enter by zakonczyc program...
```

Jak się za to zabrać? Załóżmy, że nic nie wiemy o programowaniu...

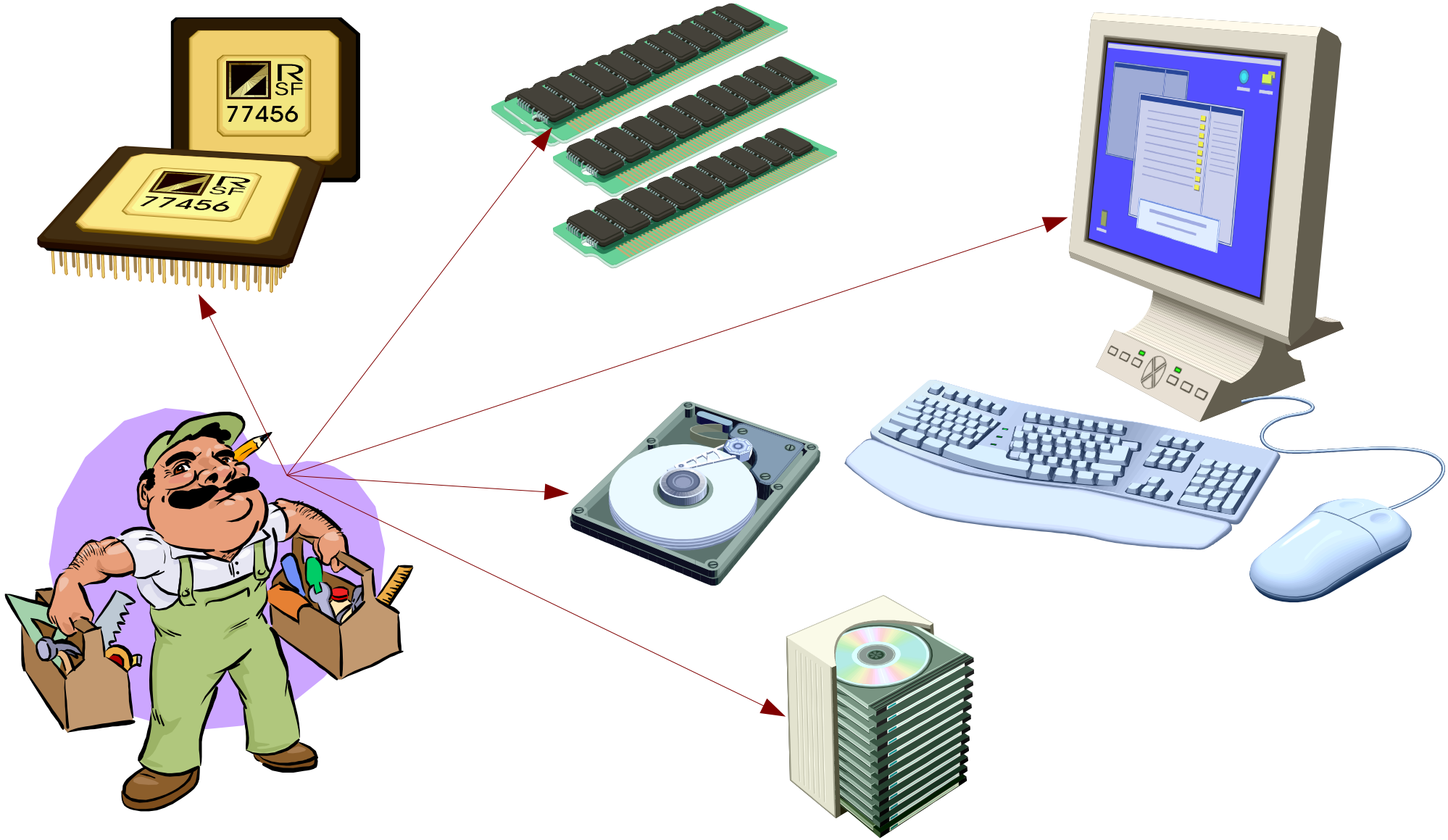
Copyright © Roman Simiński

Spojrzenie użytkownika — komputer to czarna skrzynka

Użytkownika interesuje CO potrafi zrobić komputer a nie JAK to robi… .

Copyright © Roman Simiński

Spojrzenie programisty — komputer od środka



Programista musie wiedzieć jak zmusić komputer, żeby zrobił to co trzeba.

Copyright © Roman Simiński

Algorytm, czyli JAK TO SIĘ ROBI!

Programista musi znać sposób rozwiązania problemu, który stanowi przedmiot programu. Ten sposób określa algorytm.

Algorytm określa *dane* oraz skończony ciąg *operacji*, jakie należy na tych danych wykonać, aby rozwiązać dowolny problem z określonej klasy.

Inaczej mówiąc, **algorytm** określa *dane* oraz *plan akcji*, jakie należy wykonać, aby osiągnąć założony cel.

Copyright © Roman Simiński

Cechy algorytmu

Algorytm ma być:

- *Poprawny* — dla każdego poprawnego zestawu informacji wejściowych prowadzić do poprawnych rezultatów.
- *Jednoznaczny* — każdorazowo, dla każdego poprawnego zestawu informacji wejściowych, prowadzić do tych samych rezultatów.
- *Szczegółowy* — aby *ktoś* albo *coś* wykonujący algorytm rozumiał dokładnie opisane operacje i potrafił je wykonać.
- *Uniwersalny* — pozwalał na rozwiązanie *dowolnego* problem z określonej klasy, a nie dotyczył pewnych przypadków..

Copyright © Roman Simiński

Metody zapisu algorytmów

Wykorzystuje się różne metody zapisu algorytmów:

- Język naturalny (potoczny) określający ciąg kroków algorytmu.
- Notacje graficzne, najpopularniejsze są *schematy blokowe*.
- Pseudokod — połączenie języka naturalnego i notacji matematycznej z elementami języków programowania.
- Zapis w postaci kodu programu w pewnym języku programowania.

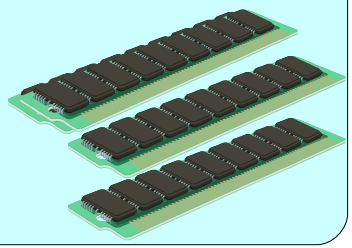
Copyright © Roman Simiński

Co to za dane, co to za operacje?

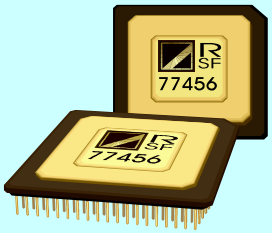
Dane, które program pamięta w czasie swojego działania

Wykonanie programu Obliczenia, sterowanie, nawroty, decyzje itp., itd. ...

Pamięć podręczna procesora

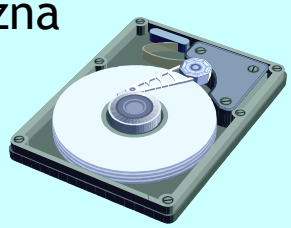


Procesor i spółka



Urządzenia wejścia i wyjścia

Pamięć zewnętrzna



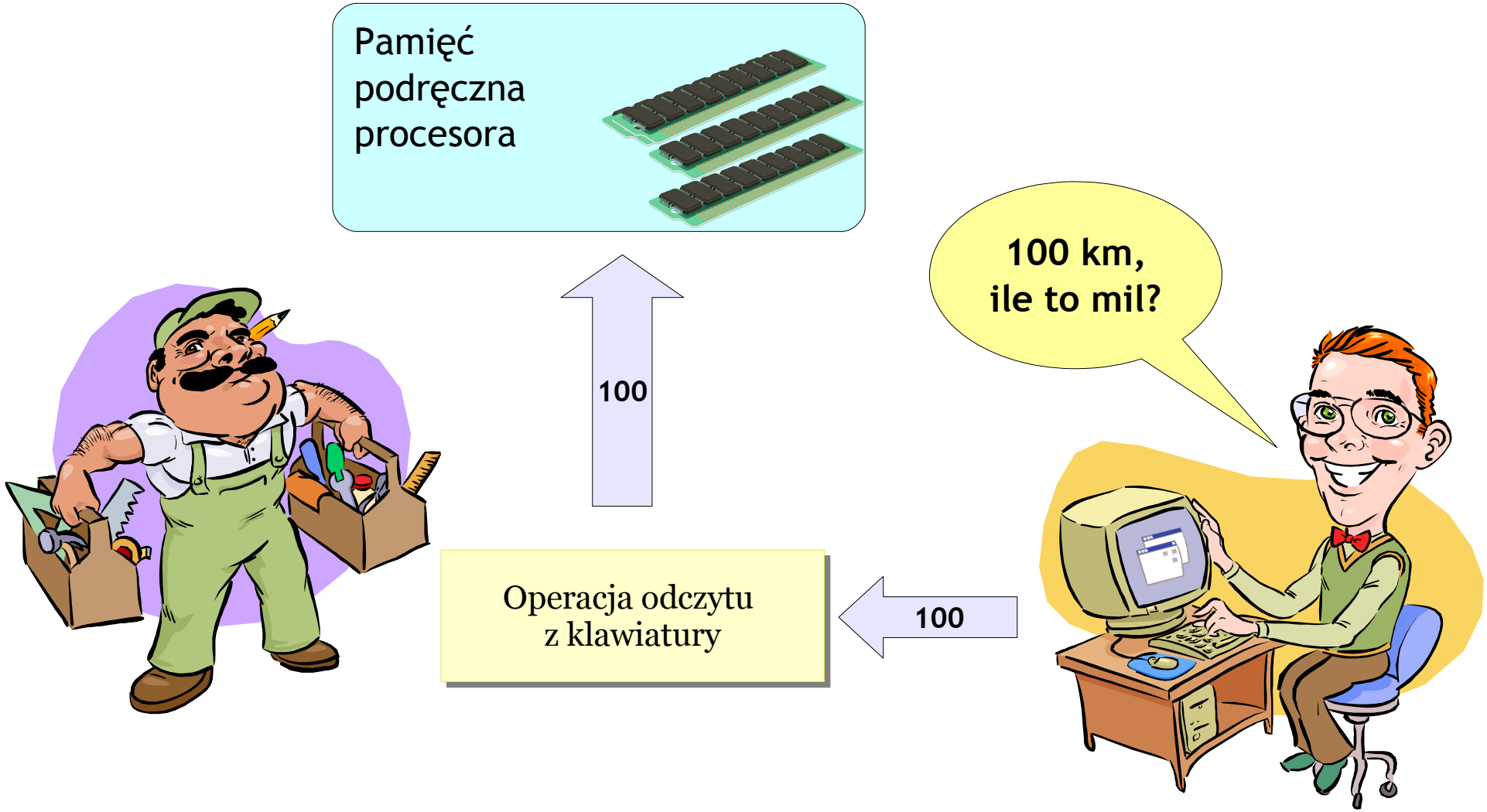
Dane, które mają być zapamiętane trwale

Operacje wejścia-wyjścia

Copyright © Roman Simiński

A co z przeliczaniem z kilometrów na mile?

Czy w tym problemie występują jakieś dane?



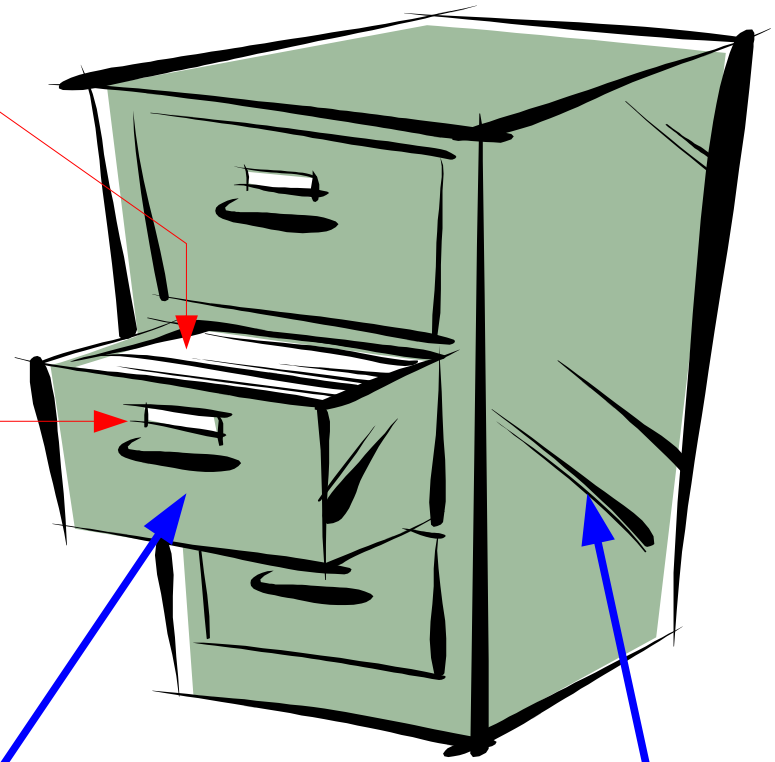
Copyright © Roman Simiński

Jak dane są przechowywane w pamięci operacyjnej?

Zmienna jest to element programu, rezydujący w pamięci operacyjnej, przeznaczony do przechowywania wartości.

Każda zmienna ma swój typ

Każda zmienna ma swoją nazwę



Zmienna

Pamięć operacyjna

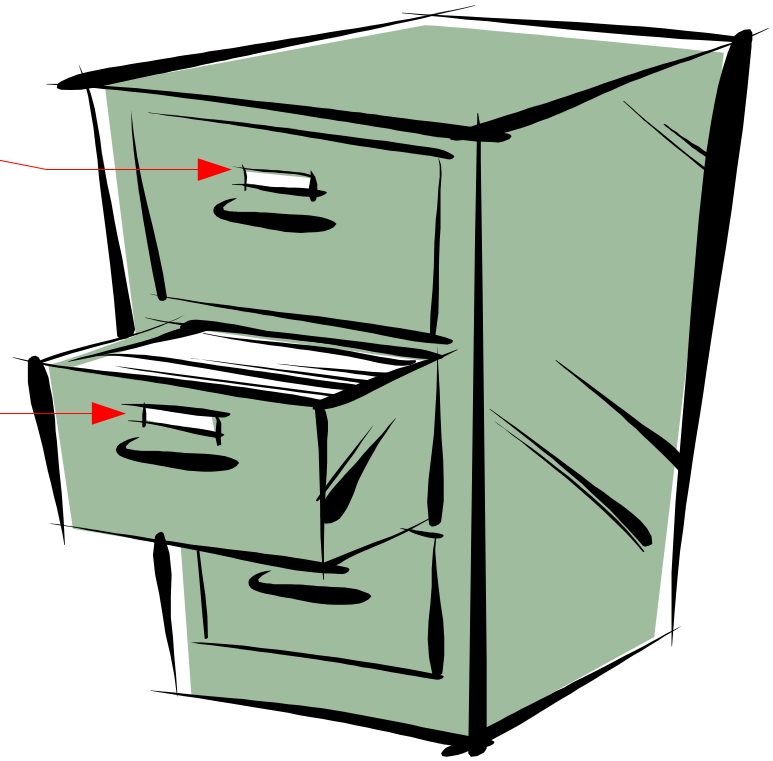
Copyright © Roman Simiński

Rola nazwy zmiennej

Nazwa zmiennej jednoznacznie identyfikuje ją w programie.
Posługując się nazwą zmiennej, odwołujemy się do zapisanej w niej wartości.

Każda zmienna ma swoją nazwę

Nazwy zmiennych powinny być unikatowe w obrębie danego fragmentu programu



Nazwa zmiennej:

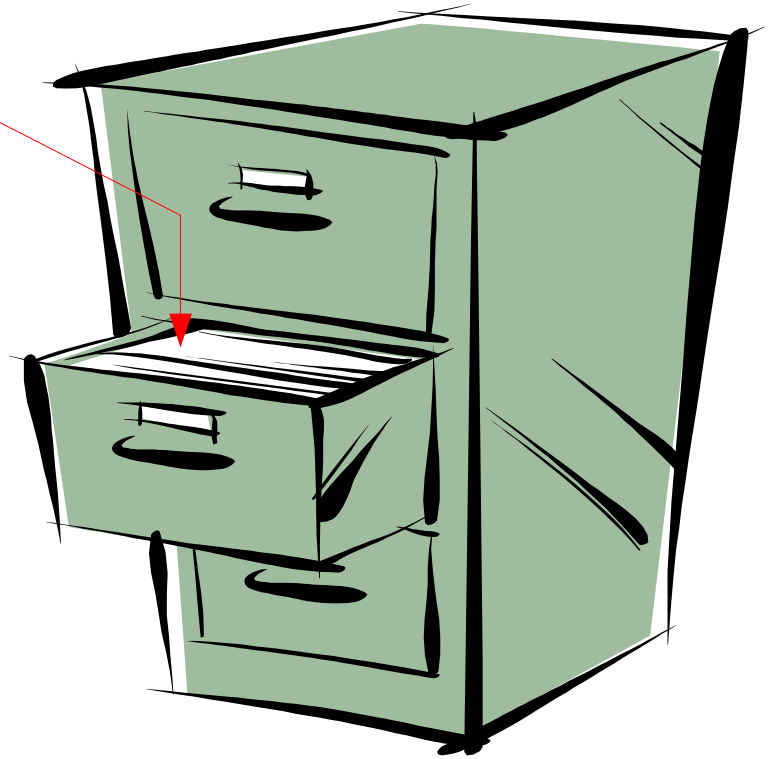
- To słowo, skrót, akronim lub ich zbitek, wymyślony przez programistę.
- Zwykle nazwa zmiennej może zawierać tylko litery, cyfry i znaki podkreślenia.

Copyright © Roman Simiński

Rola typu zmiennej

Typ zmiennej określa jakie wartości można do niej wstawiać.

Każda zmienna ma swój typ



Typ zmiennej określa dokładnie:

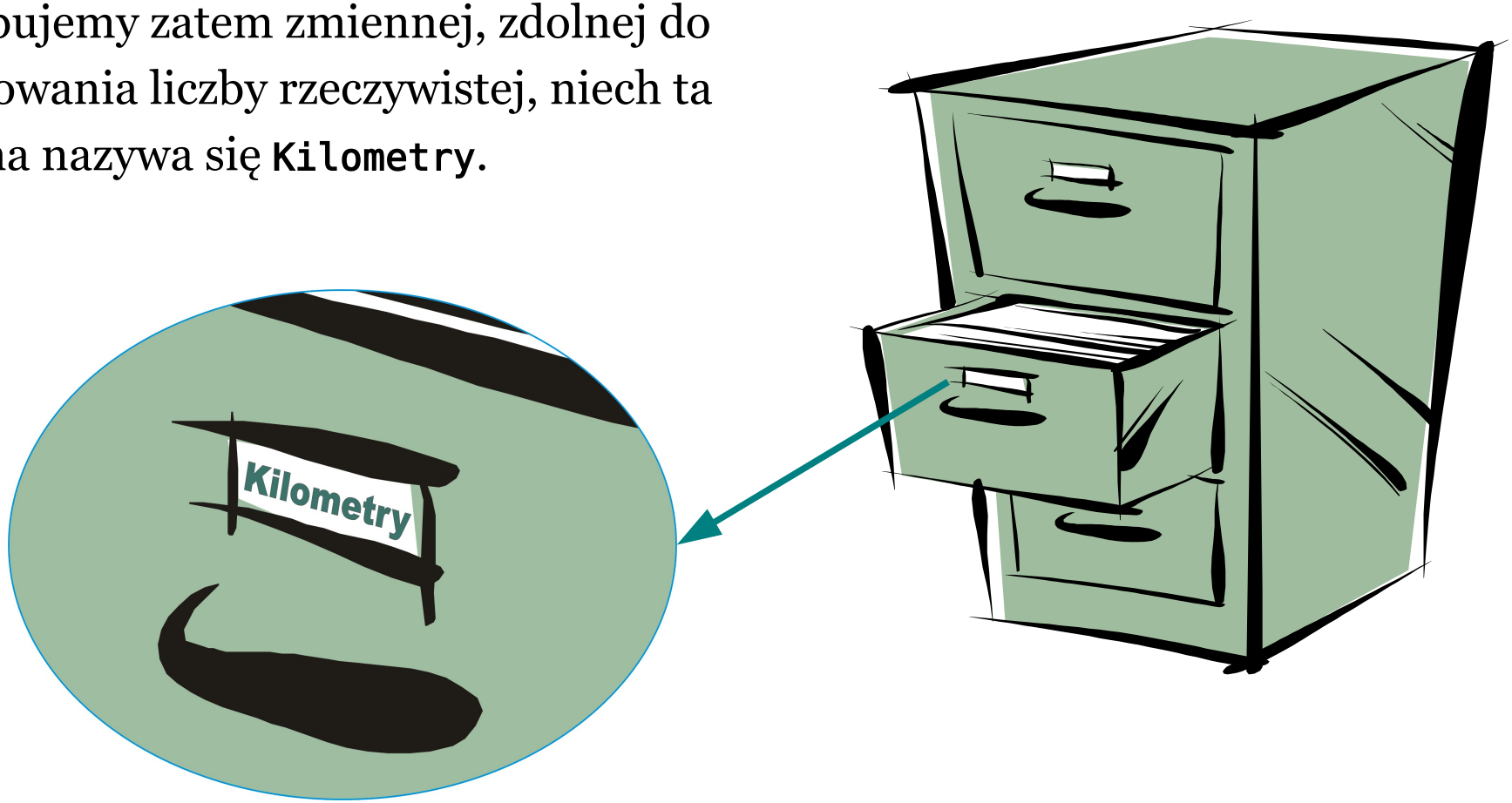
- Jakie wartości może przyjmować konkretna zmienna, i z jakiego zakresu mogą one pochodzić.
- Jakie operacje można wykonywać na zmiennej.
- Ile bajtów zmienna zajmuje w pamięci operacyjnej.

Copyright © Roman Simiński

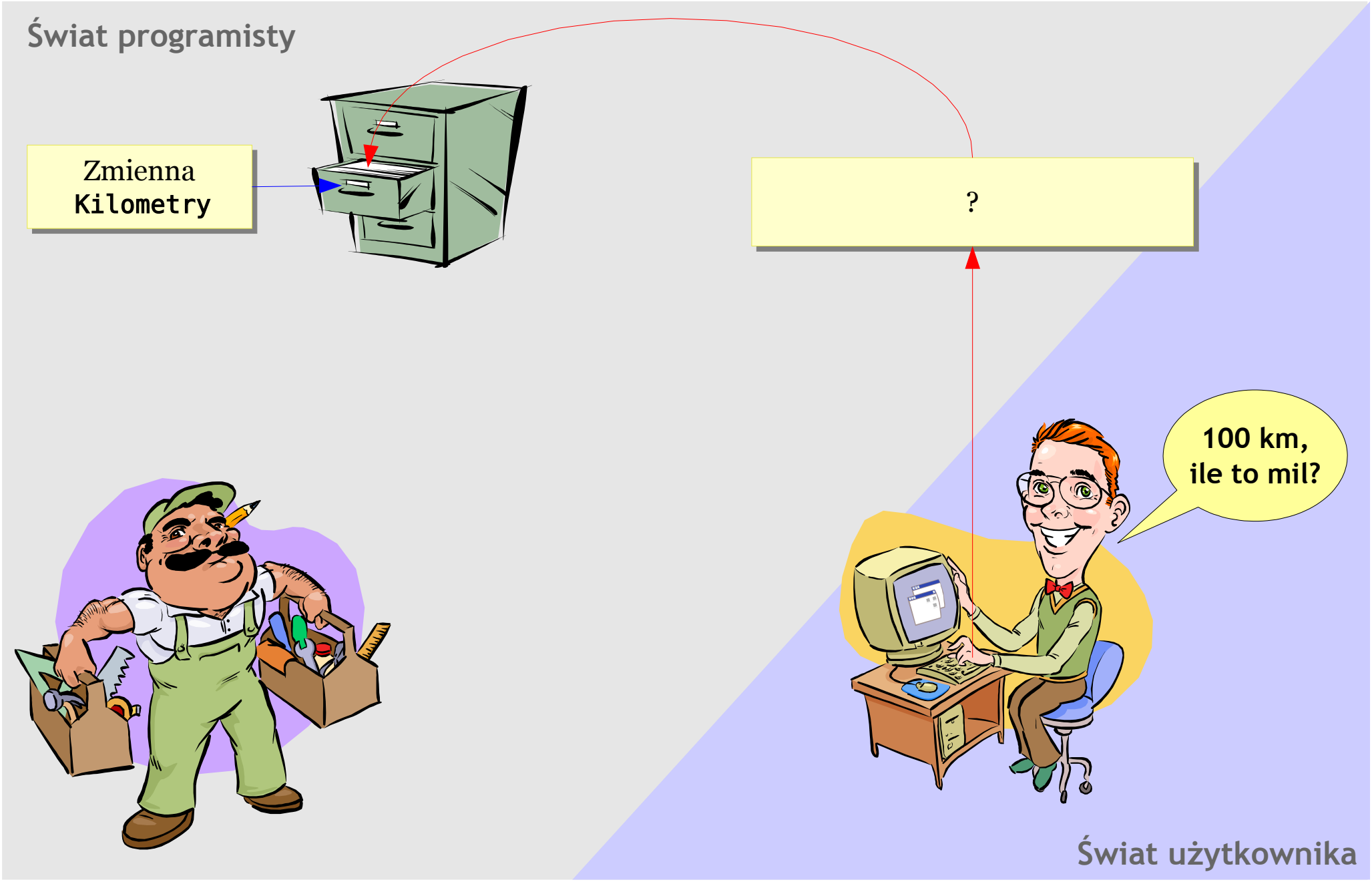
Zmienne a przeliczenie odległości

Potrzebujemy miejsca w pamięci operacyjnej, w którym przechowamy na czas działania programu odległość w kilometrach, podaną przez użytkownika.

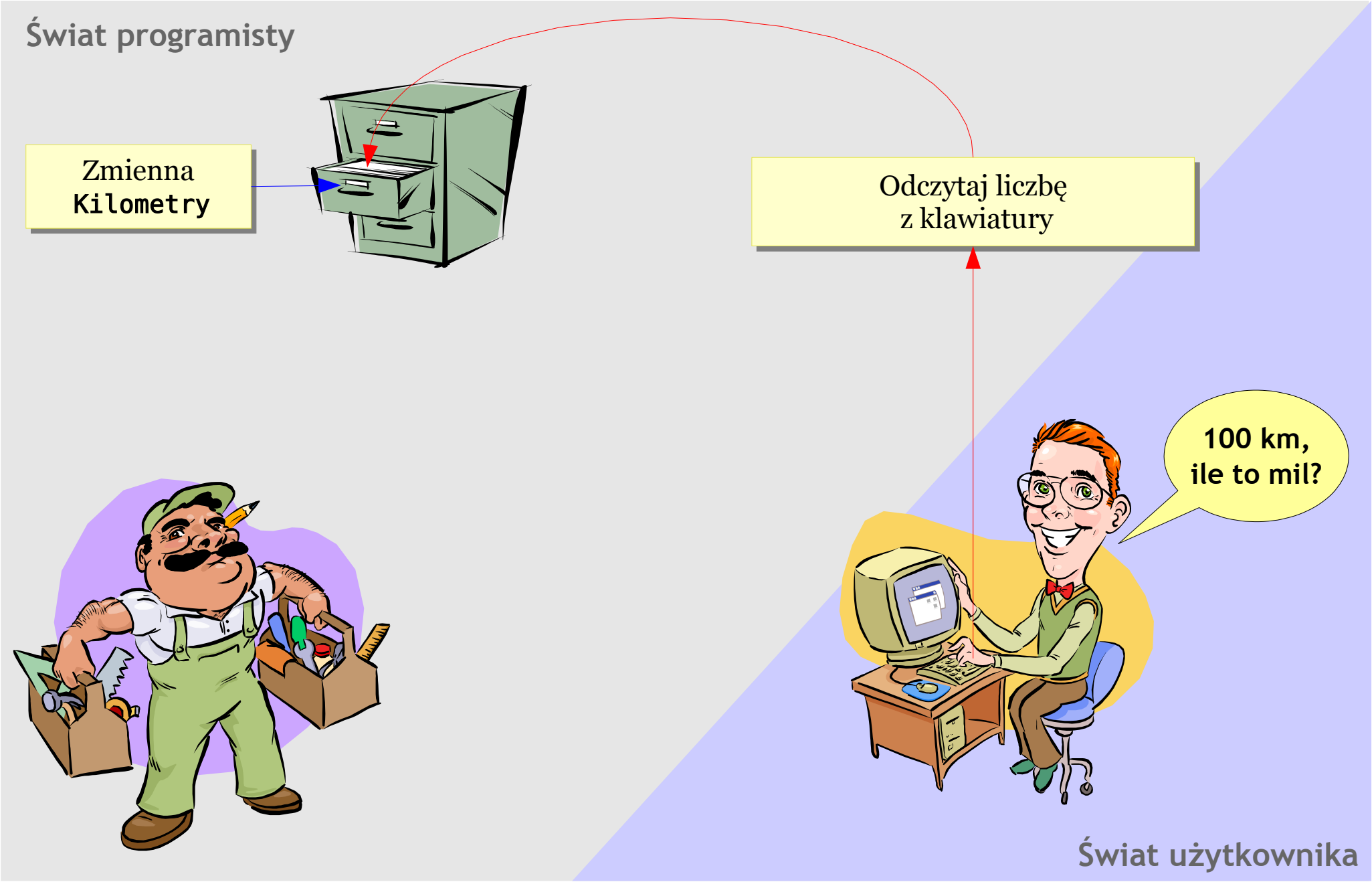
Potrzebujemy zatem zmiennej, zdolnej do przechowania liczby rzeczywistej, niech ta zmienna nazywa się `Kilometry`.



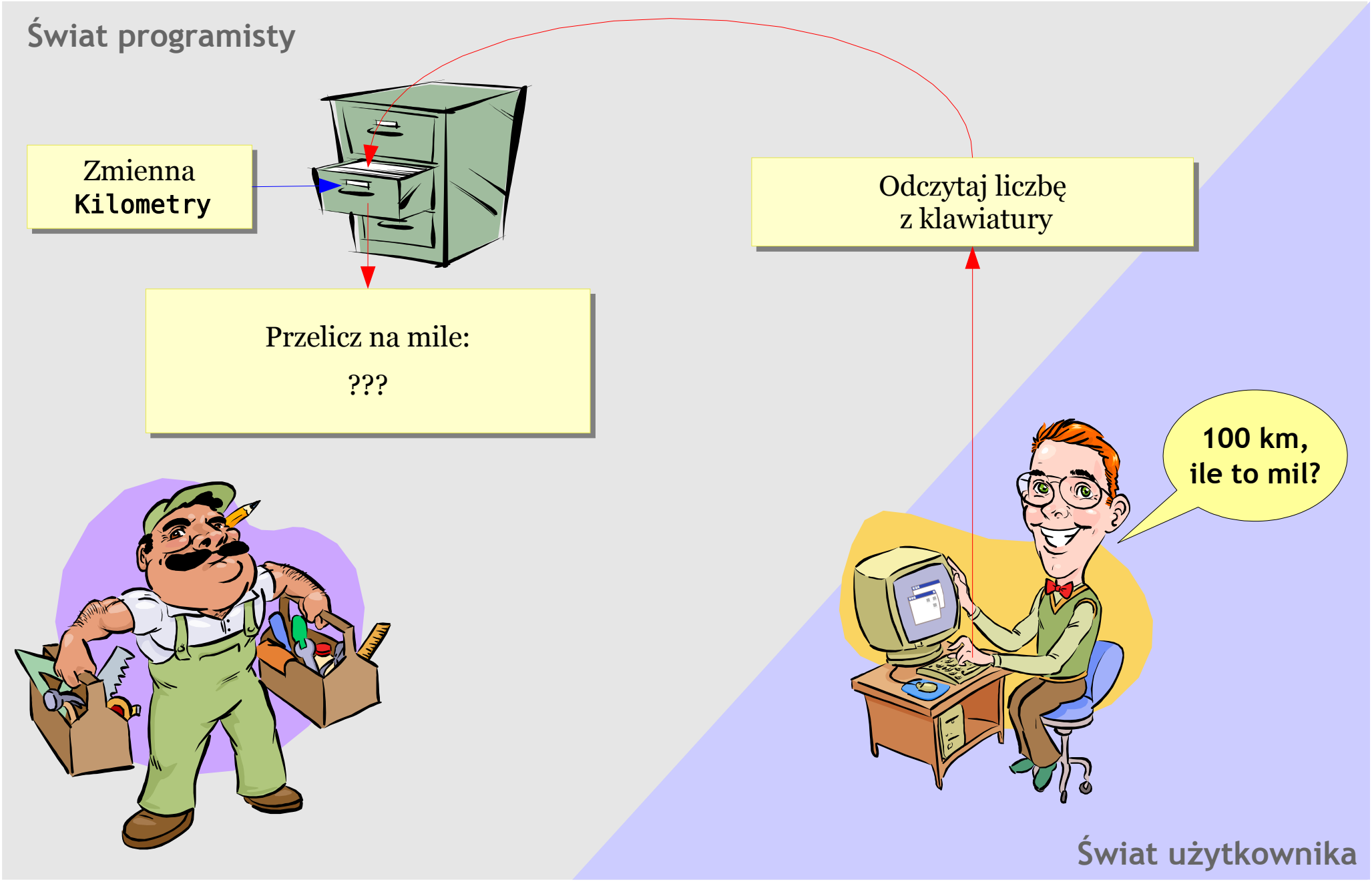
Zmienne a przeliczenie odległości



Zmienne a przeliczenie odległości



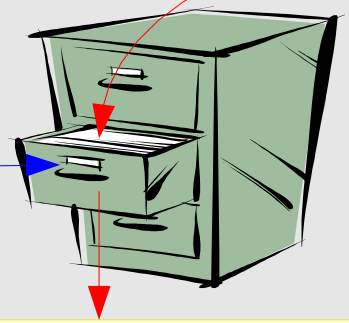
Zmienne a przeliczenie odległości



Zmienne a przeliczenie odległości

Świat programisty

Zmienna
`Kilometry`



Przelicz na mile:

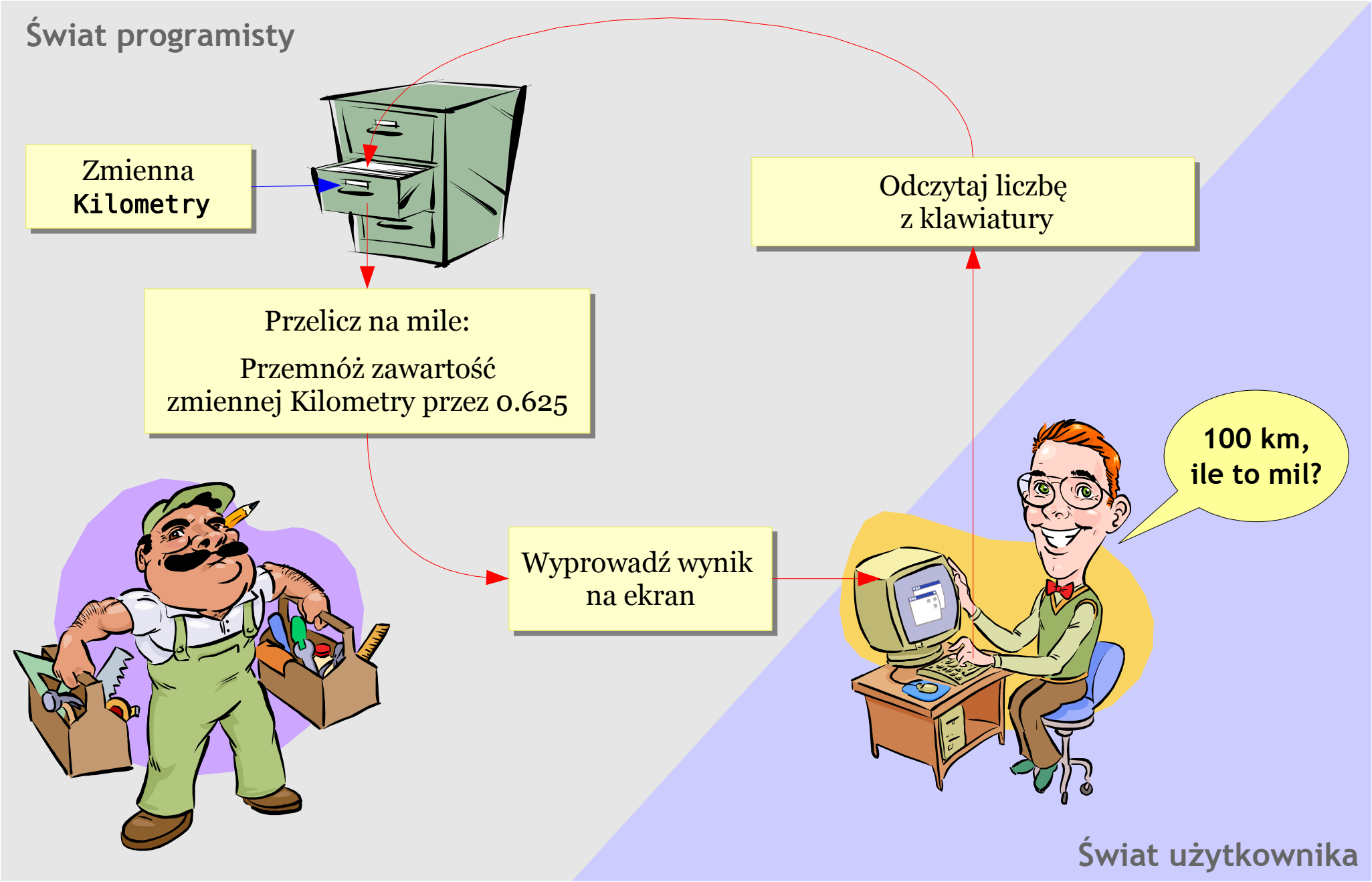
Przemnóż zawartość
zmiennej Kilometry przez 0.625

Odczytaj liczbę
z klawiatury

Świat użytkownika

Copyright © Roman Simiński

Zmienne a przeliczenie odległości



Copyright © Roman Simiński

Zmienne a przeliczenie odległości

Świat programisty

Zmienna
`Kilometry`

Odczytaj liczbę
z klawiatury

Przel

Przem
zmiennej Kil

Czy to już jest algorytm?

Tak! Tylko algorytmy zwykle się zapisuje inaczej… .

100 km,
ile to mil?

Świat użytkownika

Copyright © Roman Simiński

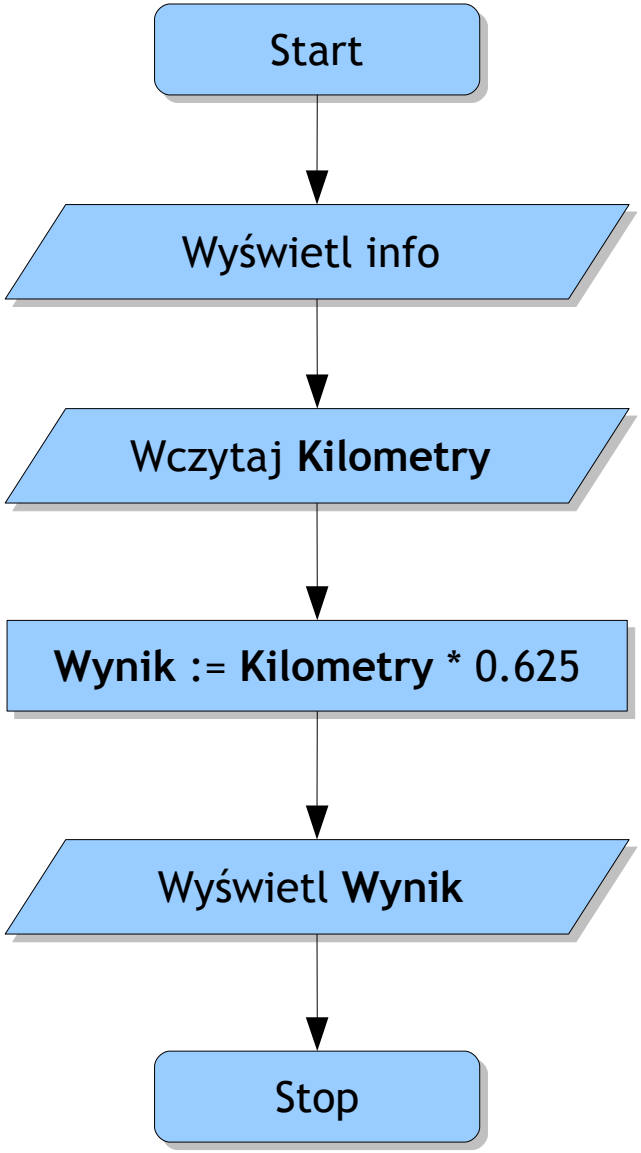
Algorytm przeliczenia odległości — opis słowny

Algorytm

- Niech **`Kilometry`** i **`Wynik`** będą zmiennymi typu liczba rzeczywista.
- Wyświetl informacje o programie.
- Wyświetl tekst zachęty do wprowadzenia liczby kilometrów.
- Odczytaj liczbę z klawiatury, zapisz w zmiennej **`Kilometry`**.
- Przelicz na mile: **`Wynik`** = zawartość zmiennej **`Kilometry`** pomnożonej przez 0.625.
- Wyprowadź **`Wynik`** na ekran.

Copyright © Roman Simiński

Algorytm przeliczenia odległości — schemat blokowy



Copyright © Roman Simiński

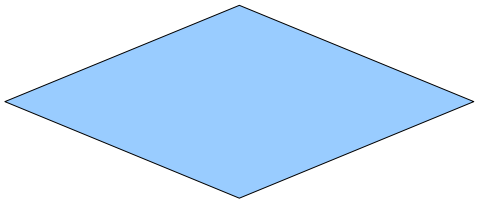
Podstawowe elementy schematów blokowych

Bloki początku i końca algorytmu:

Blok operacji wejścia/wyjścia

Blok wykonawczy

Blok decyzyjny



Copyright © Roman Simiński

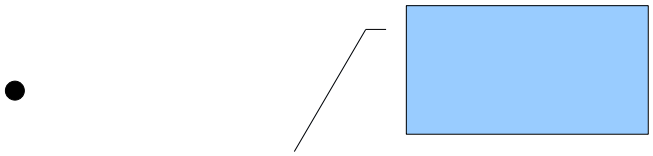
Podstawowe elementy schematów blokowych

Blok operacji wykonywanej poza rozważanym algorytmem

Łączniki stronnicowe

Łącznik międzystronnicowy

Punkt łączący i komentarz objaśniający



Copyright © Roman Simiński

Od algorytmu do programu — program, który robi nic

```
Program KilometryNaMile;
Begin
End.
```

- **Program**, **Begin**, **End** to słowa kluczowe. Słowa kluczowe to identyfikatory zastrzeżone i nie mogą być inaczej stosowane niż określa to standard języka.
- **Program** — oznacza nagłówek programu, po którym występuje identyfikator oznaczający jego nazwę. (tutaj: **KilometryNaMile**).
- **Begin** i **End** — nawiasy syntaktyczne instrukcji złożonej. Wykonanie instrukcji złożonej polega na sekwencyjnym wykonaniu jej instrukcji wewnętrznych.
- Jednomodułowy program w języku Pascal jest „zdaniem”, na końcu którego występuje znak kropki.
- Średnik — **;** — w języku Pascal jest separatorem rozdzielającym, służy do oddzielania od siebie instrukcji oraz definicji i deklaracji.

Copyright © Roman Simiński

Od algorytmu do programu — identyfikatory

```
Program KilometryNaMile;
Begin
End.
```

- Programista może pewne elementy programu nazywać. Właściwy dobór nazw wprowadza dodatkowe informacje o nazywanym obiekcie i zwiększa czytelność programu. Nazwy muszą być identyfikatorami.
- ***Identyfikator*** to ciąg liter, cyfr i znaków podkreślenia rozpoczynający się od litery, przy czym znak podkreślenia traktowany jest jako litera. Polskie znaki nie są traktowane jako litery.

Poprawne identyfikatory	Niepoprawne identyfikatory
`J23, PI, Delta, x, _y, Linia, Dlugosc`	`J 23, 2PI, 007, liczba linii, długość`

W języku Pascal nie jest istotna wielkość liter, te identyfikatory są jednakowe:

```
Liczbalinii        LICZBALINII        LICZBAlinii        LiCzBaLiNiI
```

Copyright © Roman Simiński

Od algorytmu do programu — wyprowadzanie napisów

```
Program KilometryNaMile;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );
End.
```

- **`WriteLn`** — procedura standardowa, wyprowadzająca dane do strumienia wyjściowego programu, dołączająca znacznik końca linii.
- *Napisy* (inaczej *literały łańcuchowe*) są w języku Pascal ujmowane w znaki apostrofa:

```
'Przeliczanie odleglosci wyrazonej w kilometrach na mile'
```

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
_
```

Kursor

Copyright © Roman Simiński

Od algorytmu do programu — wyprowadzanie napisów

```
Program KilometryNaMile;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );
End.
```

```
WriteLn( 'Coś tam' );
```

Wyświetl *Coś tam*, przenieś kursor do początku następnej linii

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
_
```

Kursor

```
_
```

```
WriteLn;
```

Wyświetl *nic*, przenieś kursor do początku następnej linii

Copyright © Roman Simiński

Od algorytmu do programu — zmienne

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );
End.
```

- **`Var`** — słowo kluczowe oznaczające: tutaj początek sekcji deklaracji zmiennych.
- **`Kilometry : Real`** i **`Wynik : Real`** — deklaracje zmiennych typu rzeczywistego. Deklaracja zmiennej określa jej nazwę, która musi być *identyfikatorem* oraz *typ wartości*, jakie zmienna może przechowywać.

Copyright © Roman Simiński

Od algorytmu do programu — zmienne

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );
End.
```

Real — nazwa predefiniowanego typu zmiennopozycyjnego służącego do reprezentacji liczb rzeczywistych.

- Liczby rzeczywiste w systemach komputerowych stanowią dyskretny i skończony podzbiór zbioru liczb rzeczywistych.
- Zakres wartości tego typu zależny jest od implementacji.

Copyright © Roman Simiński

Od algorytmu do programu — struktura

```
Program KilometryNaMile;

Var
  Kilometry : Real;
  Wynik     : Real;

Begin

  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

End.
```

Definicje i deklaracje

Część akcyjna programu

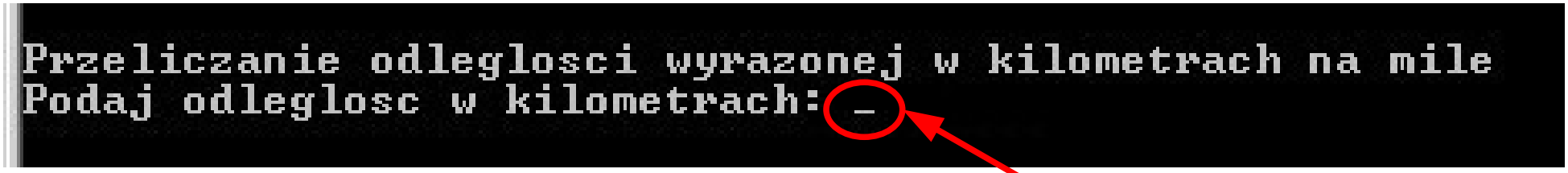
Copyright © Roman Simiński

Od algorytmu do programu — procedura Write

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
End.
```

- **Write** — procedura standardowa, wyprowadzająca dane do strumienia wyjściowego programu, kursor umieszczany jest za wyprowadzonymi informacjami.



Kursor

Copyright © Roman Simiński

Od algorytmu do programu — wprowadzanie danych

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );
End.
```

ReadLn — procedura standardowa wczytująca dane ze strumienia wejściowego programu.

- Znacznik nowego wiersza (Enter) jest pomijany.
- Dane wstawiane są do zmiennych stanowiących parametry wywołania procedury.
- Strumień wejściowy programu to zwykle klawiatura.

Copyright © Roman Simiński

Od algorytmu do programu — procedura ReadLn

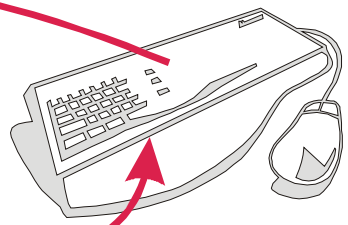
```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );
End.
```

Przed wywołaniem ReadLn:

Zawartość zmiennej **Kilometry** ???

100 + klawisz Enter



Wywołanie: **ReadLn(Kilometry);**

Po wywołaniu ReadLn:

Zawartość zmiennej **Kilometry** 100

Copyright © Roman Simiński

Od algorytmu do programu — instrukcja przypisania

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;
End.
```

:= ← Operator przypisania

Instrukcja przypisania powoduje nadanie zmiennej występującej po lewej stronie symbolu *operatora przypisania* **:=** wartości wyrażenia umieszczonego po stronie prawej.

Przed wykonaniem przypisania:

Zawartość zmiennej X	???

Przypisanie: **X := 500;**

Po wykonaniu przypisania:

Zawartość zmiennej X	500

Copyright © Roman Simiński

Od algorytmu do programu — ewaluacja wyrażenia arytmetycznego

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;
End.
```

Przed wykonaniem przypisania:

Zawartość zmiennej **Kilometry**	100

62.5

Przypisanie: **Wynik := Kilometry * 0.625;**

100 * 0.625

Po wykonaniu przypisania:

Zawartość zmiennej **Wynik**	62.5

Copyright © Roman Simiński

Od algorytmu do programu — wyprowadzanie wyniku

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;

  WriteLn( 'To w milach: ' , Wynik );
End.
```

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
Podaj odleglosc w kilometrach: 100
To w milach:  6.25000000000000E+001
```

Copyright © Roman Simiński

Od algorytmu do programu — formatowanie

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;

  WriteLn( 'To w milach: ' , Wynik : 10 : 2 );
End.
```

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
Podaj odleglosc w kilometrach: 100
To w milach:      62.50
```

Od algorytmu do programu — formatowanie, sztuczka

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;

  WriteLn( 'To w milach: ' , Wynik : 0 : 2 );
End.
```

```
Przeliczanie odleglosci wyrazonej w kilometrach na mile
Podaj odleglosc w kilometrach: 100
To w milach: 62.50
```

Copyright © Roman Simiński

Od algorytmu do programu — daj szansę zobaczyć wynik

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  WriteLn;
  WriteLn( 'Przeliczanie odleglosci wyrazonej w kilometrach na mile' );

  Write( 'Podaj odleglosc w kilometrach: ' );
  ReadLn( Kilometry );

  Wynik := Kilometry * 0.625;

  WriteLn( 'To w milach: ', Wynik : 0 : 2 );

  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

```
ReadLn( Kilometry );
```

Czekaj na liczbę, zakończ po naciśnięciu klawisza Enter

```
ReadLn;
```

Czekaj na nic, zakończ po naciśnięciu klawisza Enter

Copyright © Roman Simiński

Od algorytmu do programu — drobna optymalizacja

```
Program KilometryNaMile;
Var
  Kilometry : Real;
  Wynik     : Real;
Begin
  . . .
  Wynik := Kilometry * 0.625 ;

  WriteLn( 'To w milach: ', Wynik : 0 : 2 );
  . . .
End.
```

Zmienna **Wynik** jest zbędna (zasada przechodniości):

```
Program KilometryNaMile;
Var
  Kilometry : Real;
Begin
  . . .
  WriteLn( 'To w milach: ', Kilometry * 0.625 : 0 : 2 );
  . . .
End.
```

Copyright © Roman Simiński

Jeszcze dwa podobne przykłady

Problem

Napisać program przeliczający temperaturę wyrażona w stopniach Farenheit'a na temperaturę w stopniach Celsjusza, oraz drugi program, dokonujący operacji odwrotnej. Niech wyniki po przeliczeniu będą zaokrąglonymi liczbami całkowitymi.

Analiza

Należy odszukać formułę przeliczania pomiędzy obiema skalami temperaturowymi. Opierając się na źródłach (np. http://p609.ha.pwr.wroc.pl/chemia/miary.htm) wiemy, że:

```
temp. w st. Celsjusza = (5/9) * (temp. w st. Farenheita - 32);

temp. w st. Farenheita = (9/5) * temp. w st. Celsjusza + 32;
```

Ułamek 5/9 w wersji dziesiętnej jest okresowy, o wartości 0.555..., zatem zasadne jest pozostawienie go w takiej postaci. Ułamek 9/5 można zastąpić ułamkiem dziesiętnym 1.8.

Copyright © Roman Simiński

Stopnie Celsjusza na Fahrenheita

Scenariusz działania programu

```
Stopnie Celsjusza na Fahrenheita
Temperatura w st. C: 100
Temperatura w st. F: 212
Nacisnij Enter by zakonczyc program...
```

Kod programu

```
Program CelsjuszDoFahrenheit;
Var
  Celsjusz, Fahrenheit: Real;
Begin
  WriteLn;
  WriteLn( 'Stopnie Celsjusza na Fahrenheita' );

  Write( 'Temperatura w st. C: ' );
  ReadLn( Celsjusz );

  Fahrenheit := 1.8 * Celsjusz + 32.0;
  WriteLn( 'Temperatura w st. F: ', Round( Fahrenheit ) );

  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Stopnie Fahrenheita na Celsjusza

Scenariusz działania programu

```
Stopnie Fahrenheita na Celsjusza
Temperatura w st. F: 212
Temperatura w st. C: 100
Nacisnij Enter by zakonczyc program...
```

Kod programu

```
Program FahrenheitDoCelsjusz;
Var
  Celsjusz, Fahrenheit: Real;
Begin
  WriteLn;
  WriteLn( 'Stopnie Fahrenheita na Celsjusza' );

  Write( 'Temperatura w st. F: ' );
  ReadLn( Fahrenheit );

  Celsjusz := ( 5.0 / 9.0 ) * ( Fahrenheit - 32.0 );
  WriteLn( 'Temperatura w st. C: ', Round( Celsjusz ) );

  WriteLn( 'Nacisnij Enter by zakonczyc program...' );
  ReadLn;
End.
```

Copyright © Roman Simiński

Zaokrąglanie i obcinanie wartości rzeczywistych

Przypisywane wartości muszą być zgodne

```
Program Int2Real;
Var
  I : Integer;
  R : Real;
Begin
  R := 100;   ◄──── OK

  I := 100;
  R := I;     ◄──── OK
End.
```

```
Program Read2Int;
Var
  I : Integer;
  R : Real;
Begin
  I := 3.14;  ◄──── Błąd !

  R := 3.14;
  I := R;     ◄──── Błąd !
End.
```

`Integer` — typ całkowity, w reprezentacji komputerowej jest to podzbiór zbioru liczb całkowitych. Dozwolone operacje :

- Dodawanie (+), odejmowanie (-), mnożenie (*),
- dzielenie całkowite (**`div`**), wyznaczanie reszty z dzielenie (**`mod`**).

Dla operatorów **`div`** i **`mod`** spełnione jest:

`( m div n ) * n + ( m mod n ) = m`

Copyright © Roman Simiński

Zaokrąglanie i obcinanie wartości rzeczywistych

Wykorzystanie funkcji Round i Trunc

```
Program Read2Int;
Var
  I : Integer;
  R : Real;
Begin
  R := 3.14;
  I := Round( R );
  Write ( I );
End.
```

Zaokrąglenie

```
3
```

```
Program Read2Int;
Var
  I : Integer;
  R : Real;
Begin
  R := 3.14;
  I := Trunc( R );
  Write ( I );
End.
```

Obcięcie

```
3
```

- Funkcje **Round** i **Trunc** otrzymują jako parametr liczbę rzeczywistą, ich rezultatem jest liczba całkowita. Działają jednak w różny sposób.

Copyright © Roman Simiński

Zaokrąglanie i obcinanie wartości rzeczywistych

Działanie funkcji Round i Trunc

- **`Trunc( X )`** — rezultatem funkcji jest liczba całkowita stanowiąca część całkowitą liczby *X*.

- **`Round( X )`** — rezultatem funkcji jest liczba będąca najbliższą wartością całkowitą liczby *X*. Dla środkowej wartości przedziału przyjmuje się liczbę o większej wartości absolutnej. Zasada zaokrąglania:

```
Round( X ) = Trunc( X + 0.5 )    X >= 0

Round( X ) = Trunc( X - 0.5 )    X < 0
```

```
Round( 3.4 )=3   Round( 3.5 )=4   Round(-1.2)=-1   Round(-1.7)=-2
```

```
Trunc( 3.4 )=3   Trunc( 3.5 )=3   Trunc(-1.2)=-1   Trunc(-1.7)=-1
```

Copyright © Roman Simiński