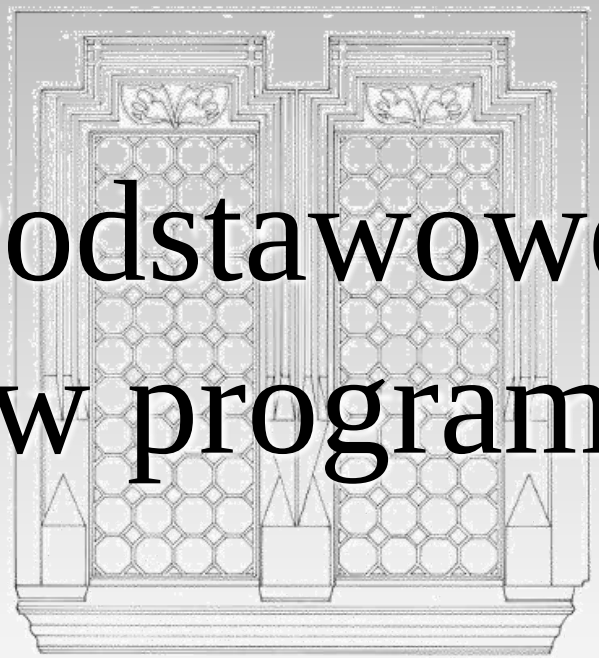

Podstawowe koncepcje w programowaniu (2)

Iteracja ograniczona

PRZYKŁAD // Zsumuj 1000 kolejnych liczb nieparzystych
// poczynając od 7

```
suma := 0;
liczba := 7;
for k := 1 to 1000 do
begin
    suma := suma + liczba;
    liczba := liczba + 2;
end
```

Iteracja ograniczona a nieograniczona

PRZYKŁAD // Zsumuj wielokrotności liczby 13 w zakresie od 26
// do 1339 włącznie

```
suma := 0;
biez := 26;
while biez <= 1339 do
begin
    suma = suma + biez;
    biez = biez + 13
end
```

Iteracja ograniczona a nieograniczona

Zdecydowanie mniej czytelne i bardziej podatne na błędy jest rozwiązanie polegające na obliczeniu najpierw ile tych liczb jest.

Pytanie: Ile jest wielokrotności liczby K nie większych do M ?
Odpowiedź: M **div** K
Pytanie: A w zakresie od N do M ?
Odpowiedź: Od liczby wielokrotności nie większych od M należy odjąć liczbę wielokrotności mniejszych od N (czyli mniejszych lub równych N–1). Zatem: (M **div** K) – ((N – 1) **div** K)

I to jest dobre ograniczenie dla zmiennej sterującej pętli **for**.

Iteracja ograniczona a nieograniczona

PRZYKŁAD // Zsumuj wielokrotności liczby 13 w zakresie od 26 // do 1339 włącznie – przy użyciu iteracji ograniczonej

```
suma := 0;
elem := 26;
for  i := 1  to  (1339 div 13) – ((26–1) div 13)  do
    begin
        suma := suma + elem;
        elem := elem + 13;
    end
```

PYTANIE

Jak powinien wyglądać algorytm dla dowolnych wartości zamiast 13, 26, 1339 ?

Zmienne tablicowe

var a: **array** [1..n] **of** typ_elementów

oznacza zadeklarowanie tablicy rozmiaru n, tj. ciągu n zmiennych tego samego typu np. integer (wtedy: **var** a: **array** [1..n] **of** integer;),

o nazwach postaci:

a[indeks_elementu]

gdzie indeks_elementu to numer elementu w zakresie od 1 do n.

Zmienne tablicowe

PRZYKŁAD

var a: **array** [1..100] **of** boolean;

deklaruje 100 zmiennych :

a[1], a[2], a[3], . . . , a[100],

które mogą przyjmować wartości logiczne.

Indeks elementu, to w takim wypadku dowolne wyrażenie arytmetyczne o wartości będącej liczbą całkowitą z zakresu [1, n].

Zmienne tablicowe

Przykładowe odwołania do elementów tablicy A:

```
 a[i];
a[2*j+1];
a[ b[k] ];    // wartości w tablicy B są indeksami dla tablicy A !
```

Zmienne tablicowe

Własności:

– Fizycznie w pamięci komputera elementy tablicy zajmują kolejne komórki pamięci.

– Zakładamy że rozmiar tablicy (tutaj: n) jest stałą.

– Wszystkie elementy tablicy są tego samego typu (wymienionego po słowie kluczowym **of**. Podstawienie pod zmienna tablicową wartości innego typu jest błędem.

– Odwołanie się do elementu poza zakresem tablicy jest błędem.

Zmienne tablicowe

Przykłady błędnego użycia zmiennych tablicowych:

```
var  b:  array [1..100] of  char; // tablica 100 znaków

k := 101;
b[k] := ‘A’;        // BŁĄD
b[k-1] := 33;      //  BŁĄD
x := b[-1];         // BŁĄD
```

Przykłady algorytmów działających na tablicach

```
var  x : array [1..100] of integer;

 // wczytywanie elementów tablicy
for  i := 1  to  100  do  read ( x[i]);

 //sumowanie elementów tablicy
suma := 0;
for  i := 1  to  100  do suma := suma + x[i];
```

Przykłady algorytmów działających na tablicach

Może też być tak:

```
//sumowanie elementów tablicy – wersja II

suma := x[1];
for  i := 2  to  100  do  suma := suma + x[i];
```

Przykłady algorytmów działających na tablicach

Podstawienie "suma := 0" przed pętlą może być w ogólnym przypadku wygodniejsze.

PRZYKŁAD
Należy zsumować k elementów tablicy d [1..100] poczynając od elementu o indeksie p (tzn. zmienna k zawiera liczbę elementów do zsumowania, zmienna p indeks elementu od którego sumowanie należy zacząć).

Zakładamy, że wartości zmiennych k i p są sensowne (tzn. odwołując się do odpowiednich elementów nie przekroczymy zakresu tablicy); dopuszczamy jednak k = 0 // !

Przykłady algorytmów działających na tablicach

PRZYKŁAD c.d.

Należy zatem zsumować elementy: x[p] , x[p+1], . . . , x[p+k–1]

```
suma := 0;
for  i := p  to   p+k–1  do
        suma := suma + x[i]              // poprawne też gdy k=0 !
```

Przykłady algorytmów działających na tablicach

```
var  x: array  [1..100] of integer;

    // sumowanie:  wersja III - z użyciem iteracji nieograniczonej
      suma := 0;
      i := 1;
      while  i <= 100  do
    begin
        suma := suma + x [i];
        i := i + 1
    end;
```

Przykłady algorytmów działających na tablicach

```
var  ch: array [1..256] of  char;
```

Oblicz ile elementów tablicy ch ma wartość 'Z'

```
ile := 0;
for  i := 1  to  256  do
        if  x [i] == 'Z'  then
                ile := ile + 1;
```

Przykłady algorytmów działających na tablicach

```
var  x: array  [1..100] of integer;
```

Znajdź wartość największego elementu w tablicy

```
max := x [1];
for  i := 2   to  100  do
        if  x [i] > max  then  max := x[i];
```

Przykłady algorytmów działających na tablicach

```
var  x: array  [1..100] of integer;
```

Znajdź indeks (czyli położenie) największego elementu w tablicy

```
i_max := 1;
for  i := 2   to  100  do
      if  x [i] > x[i_max]  then  i_max := i;

//  x[i_max] zawiera największą wartość w tablicy x
```