Podstawy Programowania

ZŁOŻONE TYPY DANYCH, LOGIKA CYFROWA I MACIERZE

Michał Bujacz

bujaczm@p.lodz.pl

B9 „Lodex” 207

godziny przyjęć: środy i czwartki 10:00-11:00

http://www.eletel.p.lodz.pl/bujacz/

Pytania z ostatniego wykładu:

- Instrukcja (Instruction)
- Zmienna (Variable), nazwa (Identifier)
- Słowo kluczowe (Keyword)
- Wyrażenie (Expression)
- Operator (Operator)
- Komentarz (Comment)

Pytania z ostatniego wykładu:

- Integer / short / long
  - co to „two’s complement” ?
- Floating point / double
  - co to mantysa i wykładnik?
- Char
- Boolean

Pytania z ostatniego wykładu:

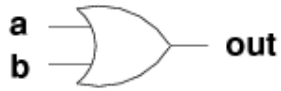
- Jaka jest największa liczba całkowita jaką możemy zapisać na 2 bajtach? Ze znakiem? Bez znaku?
- Zapisz liczbę FF34 binarnie i dziesiętnie
- Ile spacji powinno oddzielać każdy poziom kodu w Pythonie?

Logika cyfrowa

AND

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

OR

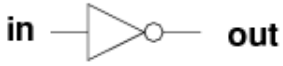


a	b	out
0	0	0
0	1	1
1	0	1
1	1	1

XOR

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

NOT



in	out
0	1
1	0

Operatory cyfrowe

x << y	przesuń bity o y w lewo po prawej wchodzą zera to samo co mnożenie przez 2^y
x >> y	przesuń bity o y w prawo po lewej wchodzą zera to samo co dzielenie przez 2^y
x & y	AND
x | y	OR
~ x	negacja bitowa liczbowo to -x - 1
x ^ y	XOR

Maskowanie bitowe

- Rejestr R

R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
?	?	?	?	?	?	?	?

- Jak sprawdzić co siedzi w R_4 ?
- Jak przestawić R_4 na 1?
- Jak przestawić R_4 na 0?

Maskowanie – odczyt bitu

- Jak sprawdzić co siedzi w R_4 ?

	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
	?	?	?	?	?	?	?	?
AND	0	0	0	1	0	0	0	0
=	0	0	0	?	0	0	0	0

```
    10011101    10010101
AND 00001000    00001000
  = 00001000    00000000
```

Maskowanie – ustawianie bitu

Jak przestawić R_4 na 1?

R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
?	?	?	?	?	?	?	?

OR	0	0	0	1	0	0	0	0
=	?	?	?	1	?	?	?	?

```
  10011101                10010101
OR 00001000             OR 00001000
 = 10011101              = 10011101
```

Maskowanie – zerowanie bitu

Jak przestawić R_4 na 0?

R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
?	?	?	?	?	?	?	?

AND	1	1	1	0	1	1	1	1
=	?	?	?	0	?	?	?	?

```
    1001 1 101    1001 0 101
AND 1111 0 111    1111 0 111
  = 1001 0 101    1001 0 101
```

Konwersja little-big endian

Jak wykonać operację x → y dla 4 bajtowych liczb?

x = 4F 52 17 01 y = 01 17 52 4F

```
y1 = x >> 24                 # 00 00 00 4F
y2 = x & 00FF0000            # 00 52 00 00
y2 = y2 >> 8                 # 00 00 52 00
y3 = x & 0000FF00            # 00 00 17 00
y3 = y3 << 8                 # 00 17 00 00
y4 = x << 24                 # 01 00 00 00
y = y1 | y2 | y3 | y4        # 01 17 52 4F
```

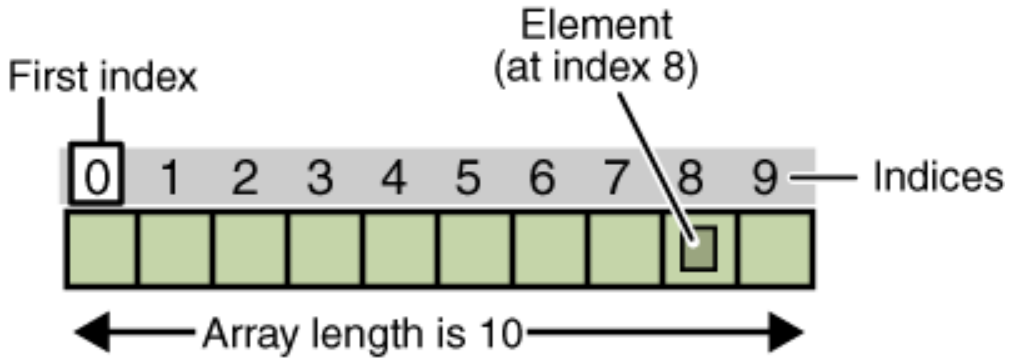
Złożone typy danych

Jedna zmienna może odnosić się do zbioru danych jednego lub kilku różnych typów

- Tablice/Sekwencje (list, str, buffer, np.matrix)
- Zbiory/słowniki (set,dict)
- Struktury/Rekordy
- Klasy/obiekty

Tablice/Sekwencje

- zmienne indeksowane, uporządkowane
- indeksy zaczynają się od 0 (wyjątki – Matlab, Mathematica, Fortran, zaczynają od 1)
- indeks podawany w kwadratowych klamrach
- zazwyczaj jeden typ danych
- jedno lub wielowymiarowe



Tablice w Pytonie

- Dość nietypowe w porównaniu z większością języków (np. Java czy C)
- mogą trzymać dowolne i różne typy danych (**heterogenous**)
- mogą dynamicznie zmieniać rozmiar i typ danych (**mutable**)
- indeksowane od 0
- można używać ujemnych indeksów

Tworzenie tablic/list

- `lista = [1, 3.14, ‘a’, ‘b’, ‘c’]`
- `lista`

`[1, 3.14, ‘a’, ‘b’, ‘c’]`

- `lista[0]`

`1`

- `lista[-1]`

`‘c’`

Tworzenie tablic (2)

- **`lista2 = [0]*5`**

#tworzy tablicę o 5 elementach równych 0

`SIZE = 1024`

`bufor = [0]*SIZE`

Metody do użycia na tablicach

<lista>.append(x)	dodaj wartość x do końca listy *(wydłuża tablicę o 1)*
<lista>.sort()	sortuj listę (parametr decyduje o kolejności)
<lista>.reverse()	odwraca kolejność elementów w liście
<lista>.index(x)	zwraca indeks kiedy wartość x pojawia się pierwszy raz
<lista>.insert(i, x)	wstawia wartość x pod indeks i *(wydłuża tablicę o 1)*
<lista>.count(x)	liczy ile razy wartość x pojawia się w liście
<lista>.remove(x)	usuwa pierwszy element równy x (*skraca tablicę o 1)*
<lista>.pop(i)	usuwa element o indeksie i, zwraca jego wartość (*skraca tablicę o 1)*

Operatory na listach

`<lista> + <lista>`	scalenie (concatenation)
`<lista> * <liczba>`	powielenie
`<lista>[i]`	wartość pod indeksem i
`len(<lista>)`	długość listy
`<lista>[:]`	cięcie (slicing) – zwraca nową listę od (włacznie) do
`for <zmienna> in <lista>:`	pętla „po liście”
`<x> in <lista>`	sprawdza czy wartość jest w sekwencji (zwraca prawdę/fałsz)

Używanie listy jako stosu

- Stos to popularna struktura programistyczna typu „Last-in, first-out”
- Odkładamy na stos - powiększamy listę o element
- Zdejmujemy ze stosu -> skracamy listę

```
> stos = [10]
> stos.append(20) # push
> stos
[10, 20]
> stos.pop() #kasuje i zwraca ostatni element
20
```

Zbiory, słowniki

Jak sekwencje, tylko nie indeksowane liczbowo

Zbiory (set) nie zawierają powtarzających się elementów

Słowniki (dict) zawierają pary klucz:wartość

Zbiory, przykład

```
>>> koszyk = ['apple', 'orange', 'apple',
  'pear', 'orange', 'banana'] #lista owocow
>>> owoce = set(basket)    #zbior owocow
>>> owoce
set(['orange', 'pear', 'apple', 'banana'])

>>> 'orange' in owoce  #czy cos jest w zbiorze?
True
>>> ‘pineapple' in owoce
False
```

Operacje na zbiorach

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                           # litery w a
set(['a', 'r', 'b', 'c', 'd'])

>>> a - b                       # litery w a, ale nie w b
set(['r', 'd', 'b'])

>>> a | b                       # litery w a lub w b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])

>>> a & b                       # litery w a i w b
set(['a', 'c'])

>>> a ^ b                       # (a | b) – (a & b)
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Przykład słownika

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}

>>> tel['jack']
4098

>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}

>>> list(tel.keys())
['irv', 'guido', 'jack']

>>> sorted(tel.keys())
['guido', 'irv', 'jack']

>>> 'guido' in tel
True

>>> 'jack' not in tel
False
```

Łańcuchy (string)

- w prezentacji Marka

Programing exercises