Zajęcia 1, 2

12 października 2025

Wstęp

Zasady zaliczenia, nieobecności, sylabus, etc.

🔗 Repozytorium — informacje i materiały z zajęć

Treści programowe

Na zajęciach omawiamy następujące zagadnienia z sylabusa:

1. Język algorytmiczny

Pojęcie zmiennej, instrukcja przypisania, instrukcje warunkowe, iteracje, operatory.

2. Pojęcie struktury tablicowej

Przykłady i implementacje prostych algorytmów dla problemów algorytmicznych na tablicach 1- i 2-wymiarowych, wyszukiwanie liniowe i binarne.

3. Pojęcie procedury

Deklaracja, parametry formalne, wywołanie, przykłady prostych procedur i funkcji.

Plan zajęć

(i) **Wprowadzenie:** Dyskusja — co to jest algorytm?

Algorytm — definicja

Skończony ciąg jasno zdefiniowanych czynności koniecznych do wykonania pewnego rodzaju zadań, sposób postępowania prowadzący do rozwiązania problemu.

Nieformalny przykład algorytmu znajdującego rzeczywiste pierwiastki równania kwadratowego:

find_quadratic_roots(a, b, c)

```
delta = find_delta(a, b, c)
if delta < 0:
   print "NO SOLUTIONS"
else:
   x1 = (-b - sqrt(delta)) / (2 * a)
   x2 = (-b + sqrt(delta)) / (2 * a)
   print "SOLUTIONS:" x1, x2
end
```

(ii) **Zmienne, instrukcje, operatory:**

- *Zmienne* to odnośniki do wartości zapisanych w pamięci komputera z przypisanymi nazwami własnymi. W większości języków programowania przypisujemy wartość do zmiennej pisząc

`x = <value>` lub `x <- <value>`

- *Instrukcje* to pojedyncze rozkazy wydawane komputerowi, które mówią, co dokładnie powinien wykonać w danym kroku działania algorytmu. Na przykład, pisząc:

```
x = 0
```

wydajemy instrukcję przypisania wartości `0` zmiennej o nazwie `x`.

- *Operatory* to po prostu skrótowe, krótkie odnośniki do najczęściej wykonywanych operacji. Na przykład zapis

```
1 + 2
```

korzysta z `+`, czyli operatora dodawania. Inne często spotykane operatory to:

```
not, or, and, -, *, /, //, >, >=, <, <=, ==
```

Zadanie 1AB

(iii) **Typy wartości i zmiennych:** W większości języków programowania, każda możliwa *wartość*, jaką możemy przypisać zmiennej, jest materializacją jednego z dostępnych *typów*. Na przykład, w języku `C`, żeby stworzyć zmienną o nazwie `x` i przypisać wartość reprezentującą *liczbę naturalną* 0, napiszemy

```
int x = 0;
```

gdzie `int` oznacza, że jest to zmienna typu *integer*. W Pythonie możemy sprawdzić typ wartości korzystając z polecenia `type()`:

```
type(1) # <class 'int'>
```

```
type(x) # <class 'int'>
```

Konkretne dostępne typy zależą od języka, z którego korzystamy. Często spotykane typy to `int`, `float` (liczba zmiennoprzecinkowa), `char` (znak), `string` (ciąg znaków), `bool` (wartość logiczna: prawda/fałsz).

Zadanie 2, Zadanie 3

(iv) **Tablice:** *Tablice* to ustrukturyzowane tabele przechowujące pojedyncze wartości. Jednowymiarowa tablica to po prostu ciąg elementów, na przykład

```
ARR1 = [1, 1, 2, 3, 5, 8, 13, 21]
```

zapisana w pamięci komputera jako pojedyncza zmienna. Dwuwymiarową tablicę możemy rozumieć jako tabelę wartości albo jako jednowymiarową tablicę przechowującą jako elementy inne jednowymiarowe tablice (tej samej długości):

```
ARR2 = [[2, 7], [1, 8], [2, 8]]
```

Do konkretnych wartości umieszczonych tablicy odwołujemy się korzystając z *indeksów*, czyli współrzędnych. Na przykład

```
ARR1[4] # == 2
```

```
ARR2[2][2] # == 8
```

Często typ `string` rozumiemy jako jednowymiarową tablicę znaków:

```
"text" ~ ['T', 'E', 'X', 'T']
```

(v) **Instrukcje warunkowe:** *Instrukcje warunkowe* pozwalają na uzależnienie wykonywanych poleceń od prawdziwości podanego warunku. Na przykład:

```
if (x > 0) and (x < 1):
    PRINT "x ∈ (0, 1)"
else if (x >= 1) and (x <= 10):
    PRINT "x ∈ [1, 10]"
else:
    PRINT "x ∈ (10, +∞)"
```

```
i = 6
is_even = bool(i // 2)
if j:
    PRINT "i is even")
else:
    PRINT "i is odd"
```

Zadanie 1C

(vi) **Pętle iteracyjne:** *Pętle* to konstrukcje programistyczne, które instruują komputer do wykonania tej samej instrukcji wiele razy, dopóki spełniony jest określony warunek albo dopóki nie skończą się dane. W językach programowania spotykamy w zasadzie dwa rodzaje pętli:

WHILE

```
x = 0
while (x < 3):
    PRINT x
    x = x + 1
```

`while` powtarza instrukcje, dopóki spełniony jest warunek definiujący pętlę

FOR

```
x = 0
for x = x + 1:
    PRINT x
    if x > 3: BREAK
```

`for` wykonuje instrukcje dla każdego elementu z danych wejściowych

Zadanie 4

(vii) **Procedury i funkcje:** *Procedura* to nazwany fragment programu — blok instrukcji, który może być wielokrotnie uruchomiony. *Funkcja* to procedura, która zwraca wartości (procedura, która nie zwraca wartości, to po prostu *procedura*):

Procedura w pseudokodzie

```
PROCEDURE ADD(i: int, j: int):
    result = i + j
PRINT result
```

Funkcja w języku Go

```
func Add(i, j int) int {
    return i + j
}
```

Zadanie 5

Następne zajęcia

Na następne zajęcia (*30 listopada 2025*) proszę o:

(i) Przygotowanie własnego środowiska do pracy w Pythonie — tak, żeby móc swobodnie uruchamiać własne skrypty i pliki `.ipynb`. Prawdopodobnie takie środowisko i tak będzie potrzebne na innych przedmiotach, więc nie powinno to wykładać sporego nakładu pracy.

(ii) Zapoznanie się z podstawami Pythona – rozsądnym testem byłaby rekreacja rozwiązań *Zadań* w Pythonie (zamiast w pseudokodzie).

(iii) Samodzielną analizę niejasności w *Zadaniach*, próbę samodzielnego rozwiązania wybranych problemów (których nie zdążyliśmy przerobić albo sprawiły Państwu kłopot).

(iv) Choćby pobieżne zapoznanie się z literaturą:
- T. Cormen, Ch. Leiserson, R. Rivest, C. Stein, *Wprowadzenie do algorytmów*: Sekcje 1.1. *Wstęp*, 1.2. *Analiza algorytmów*
- Można rzucić okiem na materiały dotyczące AiSD na stronie ważniak.mimuw.edu.pl – dość "surowe" i trudne.

Zadania

Zadanie 1. Wyznacz wartości zmiennych `i, j, k` w następujących przypadkach:

Przypadek A
```
i = 10
k = i + 6
i = 2 * k - 5
j = i + k
```

Przypadek B
```
i = 3
k = i // 2
i = h * 3
j = i / 0
```

Przypadek C
```
i, j, k = 2
j, k = i // 2, i * 2
if (k == 0) or (i // 2 == 1):
    j = j * i
```

Zadanie 2. Niech `A` i `B` będą zmiennymi liczbowymi. Napisz ciąg instrukcji "zamieniający" wartości tych zmiennych: `A` ⇆ `B`. Czy można to zrobić bez wprowadzania dodatkowej zmiennej?

Zadanie 3: Wyróżnik równania kwadratowego. Rozważmy równanie drugiego stopnia

$$ax^2 + bx + c = 0.$$

Dla danych liczb a, b, c, napisz pseudokod obliczający wyróżnik tego równania — tzw. *deltę* — dany wzorem

$$\Delta = b^2 - 4ac.$$

Jeśli a, b, c są typu `int`, to jakiego będzie typ zmiennej reprezentującej wynik?

Zadanie 4: Transpozycja macierzy. Niech A będzie macierzą o wymiarze $N \times M$ — to znaczy dwuwymiarową tablicą reprezentującą strukturę postaci

$$\begin{array}{c} \\ A[1][*] \\ A[2][*] \\ A[3][*] \end{array} \begin{array}{c} \begin{array}{ccc} A[*][1] & A[*][2] & A[*][3] \end{array} \\ \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \end{array}$$

Korzystając z pętli `while` lub `for` napisz pseudokod, który znajdzie macierz $B = A^T$, czyli dwuwymiarową tablicę `B` taką że `B[i][j] = A[j][i]` dla wszystkich $i \in [1..M]$ oraz $j \in [1..N]$.

Zadanie 5: Iloczyn skalarny.
- Napisz funkcję `DOT_PRODUCT`, która dla zmiennych wejściowych `A` i `B` (tablice liczb zmiennoprzecinkowych równej długości równej `N`) oblicza sumę iloczynów ich kolejnych wyrazów. Na przykład, dla `A = [1, 2]` i `B = [3, 4]` funkcja powinna obliczyć

```
A[1] * B[1] + A[2] * B[2] = 1 * 3 + 2 * 4 = 11
```

- Wywołaj `DOT_PRODUCT(X, X)` dla wszystkich możliwych tablic X długości $N > 0$, gdzie `X[i]` jest liczbą naturalną mniejszą od M.

Zadanie 6: Średnia arytmetyczna i harmoniczna. Niech `A` będzie tablicą liczb zmiennoprzecinkowych długości `N`. Napisz w pseudokodzie:

(A) Funkcję `arithmetic_mean(A, N)`, która oblicza średnią arytmetyczną wartości w `A`.
(B) Funkcję `harmonic_mean(A, N)` która oblicza średnią harmoniczną wartości w `A`.

Średnia harmoniczna ciągu liczbowego $X = (X_i)_{i=1}^{N}$ dana jest wzorem:

$$m_{\text{harmonic}} = N \cdot \sum_{i=1}^{N} \frac{1}{X_n}.$$

Zadanie 7: Ciąg Fibonacciego. Napisz funkcję, która dla danej liczby naturalnej `N` zwraca tablicę liczb naturalnych zawierającą pierwsze `N` wyrazów ciągu Fibonacciego. Ciąg Fibonnaciego definiujemy rekurencyjnie:

$$F_0 = 1, \qquad F_1 = 1, \qquad F_i = F_{i-2} + F_{i-1} \quad (\text{dla } i > 1).$$