Pierwsza Ławka · zestaw działowy · liceum i technikum

Dział 11

Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń

Kody ASCII i UTF-8, kod U2, zapis stało- i zmiennoprzecinkowy (IEEE 754), operacje logiczne i bitowe, przepełnienie, błędy zaokrąglenia i przybliżenia, działania w systemach innych niż dziesiętny. Osiem kart: teoria z przykładami, ćwiczenia, utrwalenie w formule CKE, zadanie przy komputerze z plikami danych (ZIP), kartkówka i sprawdzian w grupach A i B – z kluczem i zasadami oceniania.

Informatyka | klasa 1 LO i technikum | Dział 11 | zakres rozszerzony (ZR)

	Karta	Rodzaj	Poziom	Zadania
1	**Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – najważniejsze wiadomości** Teoria, przykłady krok po kroku, rozgrzewka	Najważniejsze wiadomości	ZR	4 zad. · 8 pkt
2	**Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – ćwiczenia podstawowe** Ćwiczę umiejętności z działu	Ćwiczenia	ZR	7 zad. · 17 pkt
3	**Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – utrwalenie** Zadania w formule CKE: polecenia i punktacja jak w arkuszach maturalnych	Zadania typu maturalnego	ZR	6 zad. · 10 pkt
4	**Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – w praktyce, przy komputerze** Pliki danych w archiwum ZIP · wyniki w plikach wynikiN.txt	Ćwiczenia	ZR	2 zad. · 11 pkt
5	**Kartkówka: kod U2, ASCII i operacje bitowe – grupa A** 15 minut · 8 pkt · bez komputera	Kartkówka	ZR	4 zad. · 8 pkt 15 min
6	**Kartkówka: kod U2, ASCII i operacje bitowe – grupa B** 15 minut · 8 pkt · bez komputera	Kartkówka	ZR	4 zad. · 8 pkt 15 min
7	**Sprawdzian: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – grupa A** 45 minut · 22 pkt · bez komputera	Sprawdzian	ZR	9 zad. · 22 pkt 45 min
8	**Sprawdzian: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – grupa**	Sprawdzian	ZR	9 zad. · 22 pkt 45 min

B

45 minut · 22 pkt · bez komputera

Jak korzystać z zestawu

- Karta 1 przypomina najważniejsze wiadomości i pokazuje przykłady rozwiązane krok po kroku (z kodem w Pythonie albo opisem kroków w aplikacji) – od niej warto zacząć.
- Karta 2 to ćwiczenia podstawowe, karta 3 – utrwalenie w formule zadań maturalnych CKE (ramka „Jak to sprawdza CKE").
- Karta 4 to praca przy komputerze: pliki danych są w archiwum ZIP dołączonym do zestawu, a odpowiedzi kontrolne dla plików *_przyklad (pliki o nazwie z dopiskiem „_przyklad") podano przy zadaniach. Wyniki zapisuje się w plikach wynikiN.txt – jak na maturze.
- Karty 5–6 to kartkówka (15 minut), karty 7–8 – sprawdzian (45 minut), obie w grupach A i B; te same typy zadań i ta sama punktacja, inne dane. Kartkówka i sprawdzian nie wymagają komputera.
- Klucz zawiera odpowiedzi, przykładowe programy, zasady oceniania w stylu CKE i numery wymagań podstawy. Osobny plik „Powtórka" zbiera teorię, typowe błędy i test „Sprawdź się".

Oznaczenia

ZP	zakres podstawowy
ZR	zakres rozszerzony (treści tylko dla ZR)
(0–2 pkt)	liczba punktów możliwych do zdobycia za zadanie
Zadanie 4.1.	wiązka zadań do wspólnego materiału źródłowego
Jak to sprawdza CKE	typ zadania i typowy błąd ze sprawozdań CKE
Na maturze	wymagania z podstawy programowej, których dotyczy temat

Wymagania z podstawy programowej (Dz.U. 2024 poz. 1019)

- **I.R7** przedstawia sposoby reprezentowania w komputerze znaków, liczb, wartości logicznych;
- **I.R8** objaśnia sposoby wykonywania przez komputer działań arytmetycznych i operacji logicznych;
- **I.R9** wyjaśnia, jakie może być źródło błędów pojawiających się w obliczeniach komputerowych: błąd zaokrąglenia, błąd przybliżenia;
- **I+II.R2b** wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: b) wykonywania działań na liczbach w systemach innych niż dziesiętny,

Materiał oryginalny – żadne zadanie nie jest kopią zadania CKE ani podręcznika; dane w plikach są fikcyjne. Podstawa programowa LO i technikum w brzmieniu z 2024 r. (Dz.U. 2024 poz. 1019). Wymagania: I.R7, I.R8, I.R9, I+II.R2b. Miejsce w programach nauczania: „Informatyka na czasie" (Nowa Era) ZR – część 1 (nazwy serii podano tylko informacyjnie). Do użytku własnego i pracy z własnymi uczniami.

Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – najważniejsze wiadomości

Teoria, przykłady krok po kroku, rozgrzewka

ZR </> KARTA 1/8

Imię i nazwisko: ..... Klasa: ..... Data: ..... Wynik: ..... / 8 pkt

Najważniejsze wiadomości · 8 pkt

Przypomnij sobie

- **Znaki**: każdy znak ma numer (kod). **ASCII** to kod 7-bitowy (128 znaków): 'A' = 65, 'a' = 97, '0' = 48 – mała litera ma kod o 32 większy od wielkiej. **Unicode** nadaje numery znakom wszystkich pism, a kodowanie **UTF-8** zapisuje znak na 1–4 bajtach: znaki ASCII – 1 bajt, polskie litery (ą, ż) – 2 bajty, € – 3 bajty, większość emoji (np. 🤪) – 4 bajty. W Pythonie: ord('A') = 65, chr(97) = 'a', len('ż'.encode('utf-8')) = 2.
- **Liczby naturalne na n bitach**: zakres $0 \dots 2^n - 1$ (8 bitów: 0 … 255). Wynik, który się nie mieści, powoduje **przepełnienie** – w procesorze zostaje tylko n najmłodszych bitów.
- **Kod U2 (uzupełnień do dwóch)**: najstarszy bit ma wagę **ujemną** -2^{n-1}, pozostałe – zwykłe wagi. Zakres na n bitach: $-2^{n-1} \dots 2^{n-1} - 1$ (8 bitów: −128 … 127). Np. $11001011_{U2} = -128 + 64 + 8 + 2 + 1 = -53$. Liczbę przeciwną otrzymujemy, odwracając wszystkie bity i dodając 1.
- **Zapis stałoprzecinkowy**: ustalona liczba cyfr po przecinku; wagi po przecinku to 1/2, 1/4, 1/8, … Np. $0{,}101_2 = 1/2 + 1/8 = 0{,}625$. Ułamek dziesiętny zamieniamy, mnożąc część ułamkową przez 2 i zapisując części całkowite: 0,375 → 0,75 (0) → 1,5 (1) → 1,0 (1), czyli $0{,}011_2$. Liczba 0,1 ma w systemie dwójkowym rozwinięcie **nieskończone okresowe** – nie da się jej zapisać dokładnie.
- **Zapis zmiennoprzecinkowy**: $x = m \cdot 2^c$ (znak, mantysa m, cecha/wykładnik c). Standard IEEE 754 binary64 (typ float w Pythonie): 1 bit znaku, 11 bitów wykładnika, 52 bity mantysy – 53 bity precyzji, ok. 15–16 cyfr dziesiętnych. Duży zakres, ale ograniczona dokładność: np. $2^{53} + 1$ jako float jest równe 2^{53}.
- **Wartości logiczne**: prawda = 1, fałsz = 0 (w Pythonie True/False; True + True = 2). Operacje: NOT, AND, OR, XOR (XOR = 1, gdy bity są różne). Na liczbach całkowitych działają bit po bicie: & (AND), | (OR), ^ (XOR), << i >> (przesunięcia: x << 1 = 2x, x >> 1 = x div 2).
- **Działania w komputerze**: dodawanie pisemne dwójkowe (1 + 1 = 10 – zapisujemy 0, przenosimy 1); odejmowanie a − b w U2 to dodanie a + (−b) i odrzucenie przeniesienia poza n bitów. Przepełnienie w U2: suma dwóch liczb tego samego znaku ma inny znak (np. 100 + 50 na 8 bitach daje −106).
- **Błędy obliczeń**: **błąd zaokrąglenia** – liczbę zapisujemy z ograniczoną liczbą bitów (0.1 + 0.2 daje 0.30000000000000004); **błąd przybliżenia** – zastępujemy dokładny wynik przybliżonym (np. kończymy metodę po kilku krokach, obcinamy sumę nieskończoną). Liczb float nie porównujemy przez ==, tylko z tolerancją: abs(a − b) < 1e-9 albo math.isclose(a, b).
- **Działania w innych systemach (I+II.R2b)**: dodajemy pisemnie jak w dziesiętnym, ale przenosimy 1, gdy suma cyfr osiągnie podstawę p: w systemie trójkowym $2 + 2 = 4 = 11_3$ (piszemy 1, przenosimy 1).

Przykłady z rozwiązaniem

Przykład 1. Zapisz −53 w kodzie U2 na 8 bitach i sprawdź wynik metodą wag.

$53 = 00110101_2$. Odwracamy bity: 11001010, dodajemy 1: 11001011_{U2}.

Sprawdzenie: −128 + 64 + 8 + 2 + 1 = −53.

```
x = -53
print(format(x % 256, "08b"))   # 11001011
```

Przykład 2. Dodaj w systemie trójkowym $2120_3 + 1212_3$.

Od prawej: 0 + 2 = 2; 2 + 1 = 3 = 10_3 (piszemy 0, przeniesienie 1); 1 + 2 + 1 = 4 = 11_3 (piszemy 1, przeniesienie 1); 2 + 1 + 1 = 4 = 11_3. Wynik: 11102_3 (sprawdzenie: 69 + 50 = 119 = 81 + 27 + 9 + 2).

```
def dodaj(a, b, p):          # a, b – napisy z cyframi 0..p-1 (p <= 10)
    wynik = ""
    przen = 0
    i, j = len(a) - 1, len(b) - 1
    while i >= 0 or j >= 0 or przen > 0:
        s = przen
        if i >= 0:
            s = s + int(a[i])
        if j >= 0:
            s = s + int(b[j])
        wynik = str(s % p) + wynik
        przen = s // p
        i, j = i - 1, j - 1
    return wynik

print(dodaj("2120", "1212", 3))   # 11102
```

Przykład 3. Dlaczego pętla poniżej wypisuje False? Jak to poprawić?

```
x = 0.0
for i in range(10):
    x = x + 0.1
print(x == 1.0)          # False – x = 0.9999999999999999
```

0,1 nie ma skończonego zapisu dwójkowego, więc każde dodanie wnosi mały **błąd zaokrąglenia**; po 10 krokach suma różni się od 1. Poprawki: porównanie z tolerancją **abs(x − 1.0) < 1e-9** (math.isclose), albo liczenie na liczbach całkowitych (10 · 1 dziesiąta = 10 dziesiątych).

Przykład 4. Oblicz bit po bicie 42 AND 27, 42 OR 27 i 42 XOR 27.

	bity
42	101010
27	011011
AND	001010 = 10
OR	111011 = 59
XOR	110001 = 49

```
print(42 & 27, 42 | 27, 42 ^ 27)   # 10 59 49
```

Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – ćwiczenia podstawowe

Ćwiczę umiejętności z działu

ZR </> KARTA 2/8

Imię i nazwisko: Klasa: Data: Wynik: ..... / 17 pkt

Ćwiczenia · 17 pkt

Zadanie 1. (0–3 pkt) ..... / 3

Uzupełnij tabelę – zapisz liczby w kodzie U2 na 8 bitach.

liczba	zapis U2 (8 bitów)
45	
−45	
−100	
127	

Zadanie 2. (0–3 pkt) ..... / 3

Podaj wartości dziesiętne zapisów U2.

1. 10000001_{U2} = _____ 2. 11110000_{U2} = _____ 3. 01010101_{U2} = _____

Zadanie 3. (0–2 pkt) ..... / 2

Wykonaj działania w systemie dwójkowym (wynik w systemie dwójkowym).

1. $10111_2 + 1101_2$ = _____ 2. $10111_2 - 1101_2$ = _____

Zadanie 4. (0–2 pkt) ..... / 2

Zamień.

1. 0,8125 = _____$_2$ 2. $0{,}1011_2$ = _____

Zadanie 5. (0–3 pkt) ..... / 3

Na 8 bitach w kodzie U2 komputer dodał 100 + 50. Podaj otrzymany zapis bitowy, jego wartość w U2 i nazwij zjawisko.

Miejsce na odpowiedź

Zadanie 5. (0–2 pkt) ..... / 2

Przeanalizuj program. Podaj, co wypisze, i wyjaśnij przyczynę.

```
a = 1e16
b = a + 1
print(b - a)
```

Miejsce na odpowiedź

Zadanie 6. (0–3 pkt) ..... / 3

Zapisz w pseudokodzie lub w wybranym języku programowania algorytm, który dla liczby całkowitej x (−128 ≤ x ≤ 127) zwróci 1, jeśli liczba jedynek w jej 8-bitowym zapisie U2 jest nieparzysta, albo 0 – jeśli jest parzysta (bit parzystości).

Przykład: dla x = −53 wynikiem jest 1, bo −53 = 11001011 w U2 (5 jedynek).

Uwaga: Twój algorytm może używać tylko zmiennych przechowujących pojedyncze liczby całkowite, operacji arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie całkowite i reszta z dzielenia), porównań, operatorów logicznych, instrukcji warunkowych, pętli, przypisań i własnych funkcji. Nie wolno używać funkcji wbudowanych, tablic ani list ani zamieniać liczby na napis.

Specyfikacja
Dane: x – liczba całkowita, −128 ≤ x ≤ 127
Wynik: p – 1 albo 0

Miejsce na odpowiedź

Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – w praktyce, przy komputerze

Pliki danych w archiwum ZIP · wyniki w plikach wynikiN.txt

ZR </> KARTA 4/8

Imię i nazwisko: ..... Klasa: ..... Data: ..... Wynik: ..... / 11 pkt

Ćwiczenia · 11 pkt

Zadanie 1. (0–5 pkt)

Czujnik temperatury

Opis danych

Czujnik w szklarni zapisuje co godzinę różnicę temperatur (w dziesiątych częściach stopnia) jako liczbę całkowitą w kodzie U2 na 8 bitach. Plik **czujnik.txt** zawiera 400 wierszy – w każdym 8 znaków 0/1. Pierwsze wiersze:

```
00010001
10111100
11101111
00010110
```

Plik **czujnik_przyklad.txt** zawiera 12 wierszy w tym samym formacie.

Do dyspozycji masz plik **czujnik.txt** oraz mniejszy plik **czujnik_przyklad.txt**. Odpowiedzi dla danych przykładowych podano przy zadaniach – najpierw sprawdź na nich swój program.

Napisz program (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi zapisz w pliku **wyniki1.txt**, a każdą z nich poprzedź numerem odpowiedniego zadania.

Do oceny oddajesz: plik tekstowy **wyniki1.txt** z odpowiedziami do zadań 1.1–1.3 (każda poprzedzona numerem zadania) oraz pliki z kodem źródłowym programów, np. **zadanie1.py**.

Dane fikcyjne, przygotowane do zadania.

Zadanie 1.1. (0–1 pkt) ..... / 1

Podaj, ile odczytów ma wartość ujemną. *Dla danych z pliku przykładowego odpowiedź: 4.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.2. (0–2 pkt) ..... / 2

Podaj najmniejszą i największą wartość odczytu (w systemie dziesiętnym), oddzielone spacją. *Dla danych z pliku przykładowego odpowiedź: -44 49.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 1.3. (0–2 pkt) ..... / 2

Podaj długość najdłuższego ciągu kolejnych odczytów o wartościach ujemnych. *Dla danych z pliku przykładowego odpowiedź: 1.*

Odpowiedź zapisz w pliku wyniki1.txt

Zadanie 2. (0–6 pkt)

Sumy trójkowe

Opis danych

Plik **trojkowe.txt** zawiera 200 wierszy. W każdym są dwie liczby dodatnie zapisane w systemie trójkowym (bez zer wiodących), oddzielone spacją. Pierwsze wiersze:

```
22201122 111010111
12222 12012101
202121202 221221
```

Plik **trojkowe_przyklad.txt** zawiera 10 wierszy w tym samym formacie. Sumy obliczaj w systemie trójkowym (dodawanie pisemne) – nie zamieniaj liczb na dziesiętne.

Do dyspozycji masz plik **trojkowe.txt** oraz mniejszy plik **trojkowe_przyklad.txt**. Odpowiedzi dla danych przykładowych podano przy zadaniach – najpierw sprawdź na nich swój program.

Napisz program (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi zapisz w pliku **wyniki2.txt**, a każdą z nich poprzedź numerem odpowiedniego zadania.

Do oceny oddajesz: plik tekstowy **wyniki2.txt** z odpowiedziami do zadań 2.1–2.3 (każda poprzedzona numerem zadania) oraz pliki z kodem źródłowym programów, np. **zadanie2.py**.

Dane fikcyjne, przygotowane do zadania.

Zadanie 2.1. (0–2 pkt) ..... / 2

Podaj, w ilu wierszach suma obu liczb zapisana w systemie trójkowym jest palindromem. *Dla danych z pliku przykładowego odpowiedź: 0.*

Odpowiedź zapisz w pliku wyniki2.txt

Zadanie 2.2. (0–2 pkt) ..... / 2

Podaj, w ilu wierszach zapis trójkowy sumy ma więcej cyfr niż dłuższy ze składników. *Dla danych z pliku przykładowego odpowiedź: 6.*

Odpowiedź zapisz w pliku wyniki2.txt

Zadanie 2.3. (0–2 pkt) ..... / 2

Podaj największą z sum – w systemie trójkowym. *Dla danych z pliku przykładowego odpowiedź: 1202010210.*

Odpowiedź zapisz w pliku wyniki2.txt

Zadanie 9. (0–4 pkt) ..... / 4

Zapisz w pseudokodzie lub w wybranym języku programowania algorytm, który dla liczby całkowitej x (−128 ≤ x ≤ 127) obliczy liczbę zer w jej zapisie w kodzie U2 na 8 bitach.

Przykład: dla x = −99 wynikiem jest 3.

Uwaga: Twój algorytm może używać tylko zmiennych przechowujących pojedyncze liczby całkowite, operacji arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie całkowite i reszta z dzielenia), porównań, operatorów logicznych, instrukcji warunkowych, pętli, przypisań i własnych funkcji. Nie wolno używać funkcji wbudowanych, tablic ani list ani zamieniać liczby na napis.

Specyfikacja
Dane: x – liczba całkowita, −128 ≤ x ≤ 127
Wynik: k – liczba zer w 8-bitowym zapisie U2 liczby x

Miejsce na odpowiedź

PIERWSZA ŁAWKA

CO JESZCZE JEST W ŚRODKU

Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – kartkówka, sprawdzian i karty pracy z powtórką, informatyka,...

Podgląd pokazał 9 z 29 stron. Pełna wersja: 29 stron A4, PDF do druku i DOCX do edycji.

- Karta 1: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – najważniejsze wiadomości – 4 zadania, 8 pkt
- Karta 2: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – ćwiczenia podstawowe – 7 zadań, 17 pkt
- Karta 3: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – utrwalenie – 6 zadań, 10 pkt
- Karta 4: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – w praktyce, przy komputerze – 2 zadania, 11 pkt
- Karta 5: Kartkówka: kod U2, ASCII i operacje bitowe – grupa A – 4 zadania, 8 pkt
- Karta 6: Kartkówka: kod U2, ASCII i operacje bitowe – grupa B – 4 zadania, 8 pkt
- Karta 7: Sprawdzian: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – grupa A – 9 zadań, 22 pkt
- Karta 8: Sprawdzian: Reprezentacja danych: znaki, liczby i wartości logiczne; działania i błędy obliczeń – grupa B – 9 zadań, 22 pkt Karta 1 przypomina teorię i pokazuje przykłady

Kup pełną wersję – plik od razu na e-mail

pierwszalawka.pl/material/informatyka-liceum-klasa-1-zr-dzial-11-reprezentacja-danych-znaki-liczby-i-wartosci