Rodzaj dokumentu:	**Sprawozdanie za rok 2025**
Egzamin:	**Egzamin maturalny**
Przedmiot:	**Informatyka**
Poziom:	**Poziom rozszerzony**
Termin egzaminu:	14 maja 2025 r.
Data publikacji dokumentu:	19 września 2025 r.

Sprawozdanie zostało opracowane przez Centralną Komisję Egzaminacyjną we współpracy z Okręgową Komisją Egzaminacyjną w Warszawie

Centralna Komisja Egzaminacyjna
ul. Józefa Lewartowskiego 6, 00-190 Warszawa
tel. 22 536 65 00
www.cke.gov.pl sekretariat@cke.gov.pl

Okręgowa Komisja Egzaminacyjna w Warszawie (województwo mazowieckie)
ul. Józefa Bema 87, 01-233 Warszawa
tel. 22 457 03 35
www.oke.waw.pl info@oke.waw.pl

Spis treści

Opis arkusza egzaminu maturalnego

W roku szkolnym 2024/2025 egzamin maturalny z informatyki został przeprowadzany na podstawie wymagań podstawy programowej określonych w rozporządzeniu Ministra Edukacji z dnia 28 czerwca 2024 r.[1]

Arkusz egzaminacyjny z informatyki na poziomie rozszerzonym zawierał ogółem 21 zadań w tym 19 zadań ujętych w 5 wiązkach tematycznych. Za rozwiązanie wszystkich zadań można było otrzymać 50 punktów.

Zadania sprawdzały wiadomości oraz umiejętności ujęte w trzech obszarach wymagań ogólnych:

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi

V. Przestrzeganie prawa i zasad bezpieczeństwa. Respektowanie prywatności informacji i ochrony danych, praw własności intelektualnej, etykiety w komunikacji i norm współżycia społecznego, ocena zagrożeń związanych z technologią i ich uwzględnienie dla bezpieczeństwa swojego i innych.

Egzamin trwał 210 minut. Zdający przez cały czas trwania egzaminu korzystali z autonomicznego stanowiska komputerowego.

[1] Rozporządzenie Ministra Edukacji z dnia 28 czerwca 2024 r. zmieniające rozporządzenie w sprawie podstawy programowej kształcenia ogólnego dla liceum ogólnokształcącego, technikum oraz branżowej szkoły II stopnia (Dz.U. z 2024 r. poz. 1019).

Dane dotyczące populacji zdających

TABELA 1. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZU STANDARDOWYM*

Liczba zdających (Formuła 2023)		**1 411**
Zdający rozwiązujący zadania w arkuszu standardowym	z liceów ogólnokształcących	986
	z techników	425
	z branżowych szkół II stopnia	0
	ze szkół na wsi	14
	ze szkół w miastach do 20 tys. mieszkańców	127
	ze szkół w miastach od 20 tys. do 100 tys. mieszkańców	270
	ze szkół w miastach powyżej 100 tys. mieszkańców	1 003
	ze szkół publicznych	1 192
	ze szkół niepublicznych	219
	kobiety	196
	mężczyźni	1 215
	bez dysleksji rozwojowej	1 142
	z dysleksją rozwojową	269
	obywatele Ukrainy[2]	3

* Dane w tabeli dotyczą tegorocznych absolwentów.

Z egzaminu zwolniono 15 osób − laureatów i finalistów Olimpiady Informatycznej.

TABELA 2. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZACH DOSTOSOWANYCH

Zdający rozwiązujący zadania w arkuszach dostosowanych	z autyzmem, w tym z zespołem Aspergera	79
	słabowidzący	1
	niewidomi	0
	słabosłyszący	0
	niesłyszący	0
	z niepełnosprawnością ruchową spowodowaną mózgowym porażeniem dziecięcym	0
	z zaburzeniem widzenia barw	1
	Ogółem	**81**

[2] Zdający – obywatele Ukrainy przystąpili do egzaminu maturalnego na podstawie § 3c ust. 1 rozporządzenia Ministra Edukacji i Nauki z dnia 21 marca 2022 r. w sprawie organizacji kształcenia, wychowania i opieki dzieci i młodzieży będących obywatelami Ukrainy (Dz.U. z 2023 r. poz. 2094 z późn. zm.).

Przebieg egzaminu

TABELA 3. INFORMACJE DOTYCZĄCE PRZEBIEGU EGZAMINU

Termin egzaminu			14 maja 2025
Czas trwania egzaminu dla arkusza standardowego			210 minut
Liczba szkół			233
Liczba zespołów egzaminatorów			1
Liczba egzaminatorów-weryfikatorów			1
Liczba egzaminatorów			18
Liczba obserwatorów[3] (§ 8 ust. 1)			3
Liczba unieważnień[4]	w przypadku:		
	art. 44zzv pkt 1	stwierdzenia niesamodzielnego rozwiązywania zadań przez zdającego	0
	art. 44zzv pkt 2	wniesienia lub korzystania przez zdającego w sali egzaminacyjnej z urządzenia telekomunikacyjnego	0
	art. 44zzv pkt 3	zakłócenia przez zdającego prawidłowego przebiegu egzaminu	0
	art. 44zzw ust. 1	stwierdzenia podczas sprawdzania pracy niesamodzielnego rozwiązywania zadań przez zdającego	0
	art. 44zzy ust. 7	stwierdzenie naruszenia przepisów dotyczących przeprowadzenia egzaminu maturalnego	1
	art. 44zzy ust. 10	niemożność ustalenia wyniku (np. zaginięcie karty odpowiedzi)	0
Liczba wglądów[4] (art. 44zzz)			78

[3] Rozporządzenie Ministra Edukacji Narodowej z dnia 1 sierpnia 2022 r. w sprawie egzaminu maturalnego (Dz.U. poz. z 2024 poz. 302, z późn. zm.).

[4] Ustawa z dnia 7 września 1991 r. o systemie oświaty (Dz.U. z 2025 r. poz. 881.).

Podstawowe dane statystyczne

Wyniki zdających

WYKRES 1. ROZKŁAD WYNIKÓW ZDAJĄCYCH

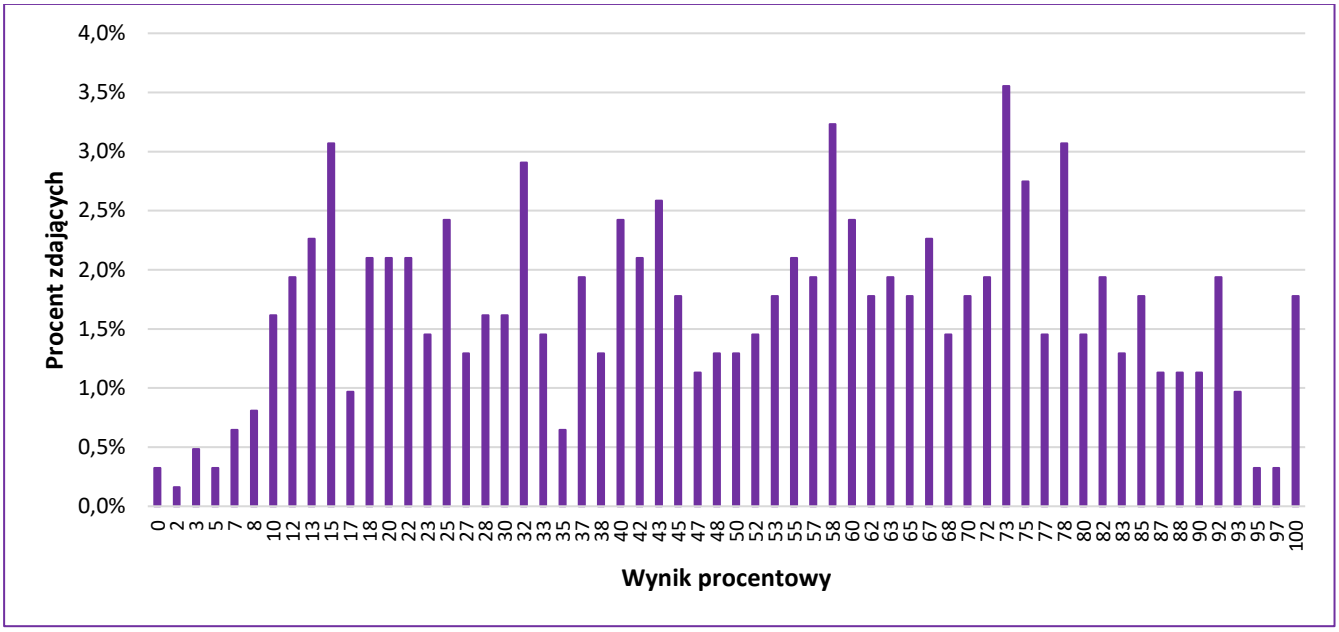


TABELA 4. WYNIKI ZDAJĄCYCH – PARAMETRY STATYSTYCZNE*

*

Zdający	Liczba zdających	Minimum (%)	Maksimum (%)	Mediana (%)	Modalna (%)	Średnia (%)	Odchylenie standardowe (%)
ogółem	**1 411**	**0**	**100**	**44**	**18**	**47**	**30**
w tym:							
z liceów ogólnokształcących	986	0	100	52	100	52	30
z techników	425	0	100	28	18	35	25

Dane dotyczą wszystkich tegorocznych absolwentów. Parametry statystyczne są podane dla grup liczących 30 lub więcej zdających.

Poziom wykonania zadań

TABELA 5. POZIOM WYKONANIA ZADAŃ

Wymagania podstawy programowej			
Nr zad.	**Wymagania ogólne**	**Wymagania szczegółowe** *Gdy wymaganie dotyczy treści zakresu podstawowego szkoły ponadpodstawowej – dopisano (P).*	**Poziom wykonania zadania (%)**
1.1.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. I+II.3) objaśnia oraz porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przykłady problemów i algorytmów, w szczególności: b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali). P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.	66
1.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. I+II.3) objaśnia oraz porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przykłady problemów i algorytmów, w szczególności: b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali). P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.	72
1.3.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. I+II.1) zapisuje za pomocą listy kroków lub pseudokodu i implementuje w wybranym języku programowania algorytmy […]. 3) objaśnia porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przykłady problemów i algorytmów, w szczególności: b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali). P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin […].	25
2.1.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;	57

		I.5) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność; I.7) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. I+II.1) zapisuje za pomocą listy kroków lub pseudokodu i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach […]. P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara. P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	
2.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych, w szczególności programuje algorytmy z punktu I.2). II.1) projektuje i tworzy programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur; II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.	29
2.3.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach: […], zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, […]. P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych, w szczególności programuje algorytmy z punktu I.2). II.1) projektuje i tworzy programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur; II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.	45

2.4.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach: […], zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, […]. P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych, w szczególności programuje algorytmy z punktu I.2). II.1) projektuje i tworzy programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur; II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.	40
3.1.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych. P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach [...]. P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	42
3.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych. P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach [...]. P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	33

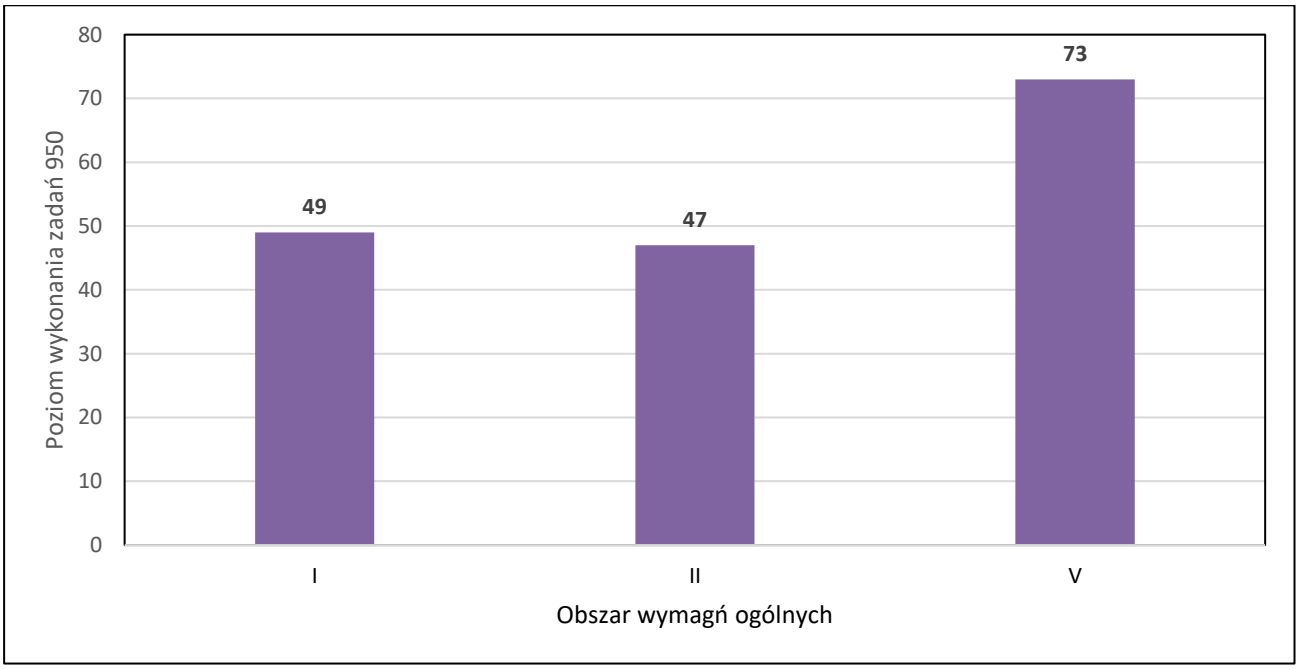
4.	V. Przestrzeganie prawa i zasad bezpieczeństwa.	Zdający: P.V.3) stosuje dobre praktyki w zakresie ochrony informacji wrażliwych (np. hasła, pin), danych i bezpieczeństwa systemu operacyjnego, objaśnia rolę szyfrowania informacji; P.V.4) opisuje szkody, jakie mogą spowodować działania pirackie w sieci, w odniesieniu do indywidualnych osób, wybranych instytucji i całego społeczeństwa.	73
5	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach: […] zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi […]. I+II.2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: b) wykonywania działań na liczbach w systemach innych niż dziesiętny.	80
6.1.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	59
6.2.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	72

6.3.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	52
6.4.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	59
6.5.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	40
7.1.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do	62

		wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	
7.2.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	42
7.3.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	35
7.4.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	31

7.5.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie. P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami: d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	41

WYKRES 2. POZIOM WYKONANIA ZADAŃ W OBSZARZE WYMAGAŃ OGÓLNYCH



Komentarz

W roku 2025 do egzaminu maturalnego z informatyki w Formule 2023 przystąpili po raz trzeci absolwenci liceów ogólnokształcących, a po raz drugi – absolwenci techników. Egzamin odbył się tylko na poziomie rozszerzonym. Średni wynik, jaki osiągnęli wszyscy absolwenci (liceów oraz techników łącznie), wynosi **43%**. Absolwenci liceów osiągnęli średni wynik **50%**, natomiast absolwenci techników osiągnęli średni wynik **34%**.

Analiza jakościowa zadań

Arkusz składał się z 21 zadań. 19 zadań ujęto w 5 wiązek tematycznych, pozostałe dwa były zadaniami odrębnymi. 11 zadań było dla zdających trudnych (poziom wykonania[5] każdego z tych zadań mieścił się w zakresie od 20% do 49%), 6 zadań okazało się umiarkowanie trudnymi (poziom wykonania w zakresie od 50% do 69%), cztery zadania były dla zdających łatwe (poziom wykonania w zakresie od 70% do 89%). W arkuszu nie było zadań bardzo trudnych (o poziomie wykonania mniejszym lub równym 19%) ani bardzo łatwych (o poziomie wykonania powyżej 89%).
Rozkład punktacji na poszczególnych poziomach trudności wyglądał następująco:

- za zadania trudne można było uzyskać łącznie 32 punkty (64% maksymalnej liczby punktów do zdobycia),
- za zadania umiarkowanie trudne można było uzyskać łącznie 15 punktów (30% maksymalnej liczby punktów do zdobycia),
- za zadania łatwe można było uzyskać łącznie 3 punkty (6% maksymalnej liczby punktów).

Zdecydowaną większość zadań stanowiły zadania praktyczne – wymagające użycia komputera. Za ich rozwiązanie można było uzyskać łącznie 36 punktów, czyli 72% wszystkich możliwych punktów do zdobycia. Za poprawne rozwiązanie każdego z tych zadań zdający otrzymywali punkty tylko wówczas, gdy zapisana odpowiedź w pliku tekstowym wynikała z komputerowej realizacji zadania. Brak plików zawierających komputerową realizację zadań był jednoznaczny z brakiem rozwiązania tych zadań.

Zadania, z którymi zdający poradzili sobie najsłabiej

W tej części komentarza omówimy zadania, z którymi zdający poradzili sobie najsłabiej. Przyjmiemy do analizy zadania, których poziom wykonania jest niższy niż 35%. Takich zadań w arkuszu jest 6.

Zadania, których poziom wykonania jest niższy niż 41%, wymieniając kolejno od najtrudniejszego, to (w nawiasach oprócz poziomu wykonania zapisano jakiej tematyki dotyczy zadanie oraz rodzaj zadania):

---

[5] Poziom wykonania zadania (lub grupy zadań) to parametr, który określamy jako iloraz (wyrażony w procentach) średniego wyniku za dane zadanie (lub grupę zadań) i maksymalnej liczby punktów możliwych do uzyskania za to zadanie (lub grupę zadań).

1. Zadanie 1.3. (25%, zadanie z zakresu algorytmiki – wymagające zapisu algorytmu – otwarte),
2. Zadanie 2.2. (24%, zadanie programistyczne, praktyczne),
3. Zadanie 7.4. (31%, zadanie bazodanowe, praktyczne),
4. Zadanie 3.2. (33%, zadanie programistyczne, praktyczne),
5. Zadanie 7.3. (35%, zadanie bazodanowe, praktyczne),
6. Zadanie 2.4. (40%, zadanie programistyczne, praktyczne).

Wśród zadań najtrudniejszych, podobnie jak w ubiegłych latach, znalazły się zadania wymagające zapisania algorytmu lub programu. W tym roku najtrudniejszymi dla maturzystów zadaniami okazały się także dwa zadania bazodanowe, które wymagały utworzenia kwerend.

Zadanie 1.3. (Rysunek 1.) jest trzecim zadaniem w wiązce zadań o nazwie „Funkcja rekurencyjna”. W zadaniu tym należało napisać w arkuszu egzaminacyjnym, w postaci pseudokodu lub w wybranym (w deklaracji maturalnej) języku programowania, nierekurencyjną (iteracyjną) wersję podanej i analizowanej na początku wiązki zadań funkcji *przestaw(n)*. Zadanie to okazało się najtrudniejszym dla zdających w tegorocznym arkuszu maturalnym z informatyki.

Zadanie 1.3. (0–4)

W postaci pseudokodu lub w wybranym języku programowania napisz **nierekurencyjną** funkcję *przestaw2*, która dla danej nieujemnej liczby całkowitej *n* da taką samą wartość jak *przestaw*(*n*).

Uwaga: Twój algorytm może używać **wyłącznie zmiennych przechowujących liczby całkowite** oraz może operować **wyłącznie na liczbach całkowitych**. W zapisie możesz wykorzystać tylko operacje arytmetyczne: dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, resztę z dzielenia oraz porównywanie liczb, instrukcje sterujące, przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje. **Zabronione** jest używanie funkcji wbudowanych oraz operatorów innych niż wymienione, **w tym – funkcji *przestaw***.

Specyfikacja:

Dane

n – nieujemna liczba całkowita

Wynik

w – nieujemna liczba całkowita, wynik działania taki sam jak po wykonaniu *przestaw*(*n*)

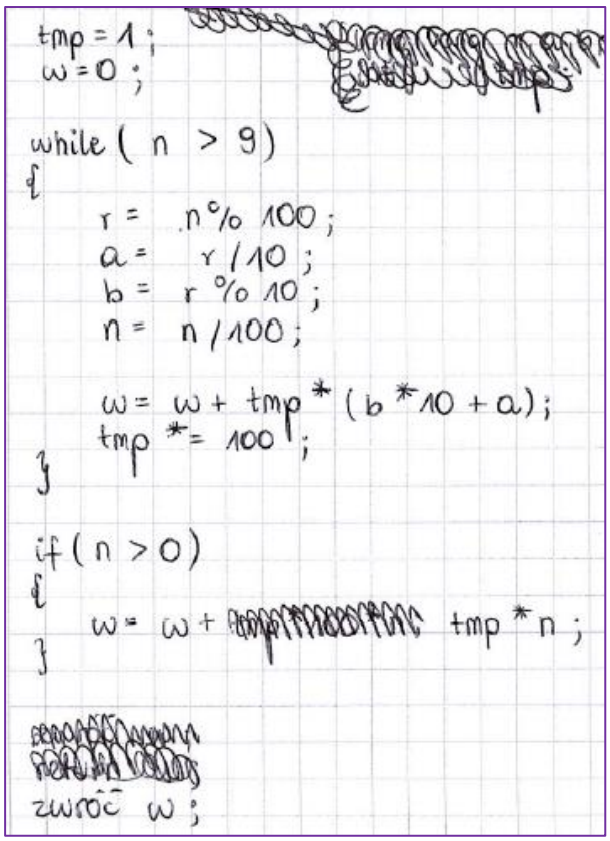
Rysunek 1. Zadanie 1.3.

Zadania tego typu często sprawiają trudności zdającym. Sytuacja ta nie zmieniła się, mimo zmian wprowadzonych w 2023 roku, w wyniku których zdający pisząc algorytm mogą wspomóc się komputerem dostępnym podczas całej pracy z arkuszem (w formule 2015 tego typu zadanie pojawiało się w części pierwszej egzaminu, podczas której nie było dostępu do komputera). Zdający mogą także od 2023 roku tak rozplanować czas przeznaczony na inne zadania, żeby mieć go więcej na zapis algorytmu. Można odnieść wrażenie, że możliwość zapisania algorytmu jako programu i sprawdzenia jego działania na komputerze

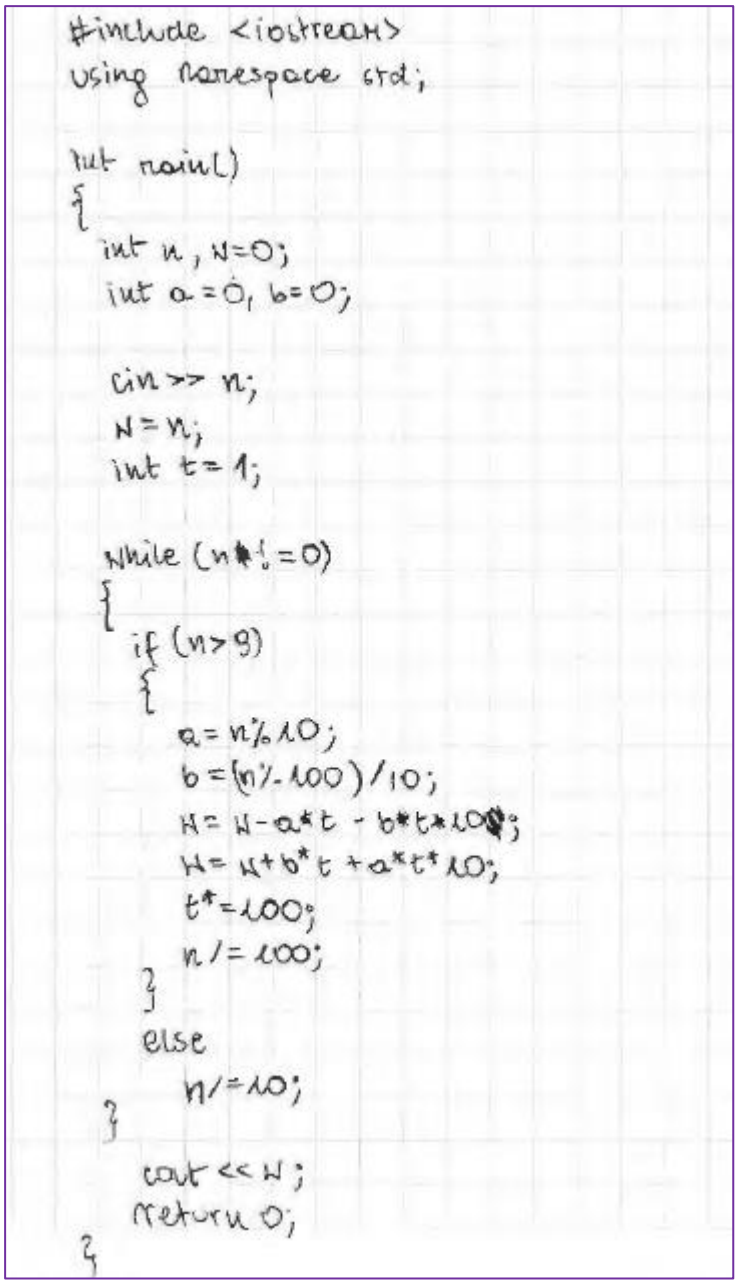
niekoniecznie stanowi dla zdających pomoc, a w niektórych przypadkach może wręcz stanowić przeszkodę w zapisaniu poprawnego algorytmu, z uwzględnieniem warunków zapisanych w uwadze pod poleceniem z zadania.

W tym przypadku zdający mieli do dyspozycji wersję rekurencyjną funkcji, z której mogli zaczerpnąć niektóre elementy, np. sposób zamiany miejscami cyfr w liczbie dwucyfrowej, sposób pobierania ostatniej cyfry, otrzymania liczby dwucyfrowej złożonej z dwóch ostatnich liczb w zapisie, itd. Głównym problemem do rozwiązania dla zdających było pobieranie kolejnych cyfr i tworzenie liczby wynikowej bez wykorzystania rekurencji.

Przykładowe **poprawne** rozwiązania zdającego przedstawiono na Rysunkach 2.–5.:



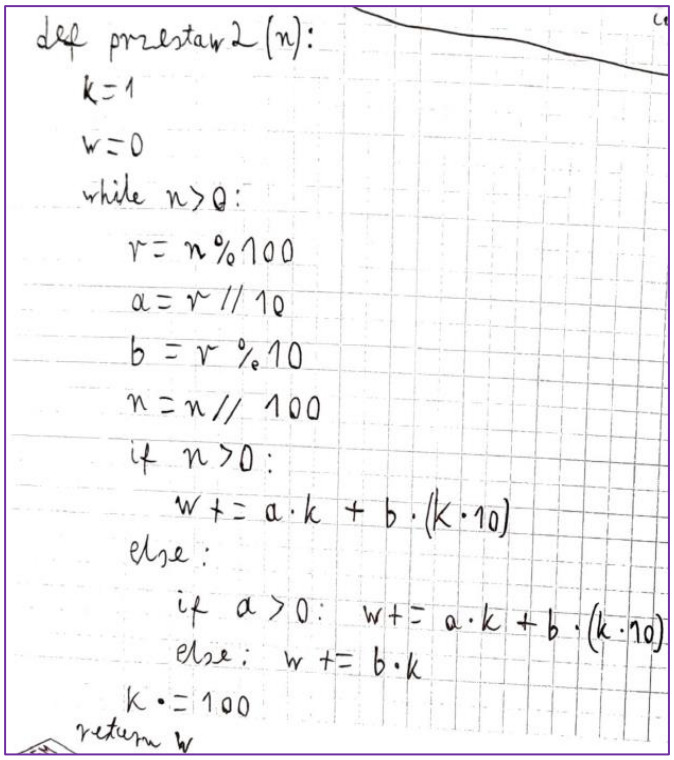
Rysunek 2. Rozwiązanie 1, c++



```
#include <iostream>
using namespace std;

int main()
{
  int n, w=0;
  int a=0, b=0;

  cin >> n;
  w = n;
  int t=1;

  while (n!=0)
  {
    if (n>9)
    {
      a = n%10;
      b = (n%100)/10;
      w = w - a*t - b*t*10;
      w = w + b*t + a*t*10;
      t*=100;
      n/=100;
    }
    else
      n/=10;
  }

  cout << w;
  return 0;
}
```

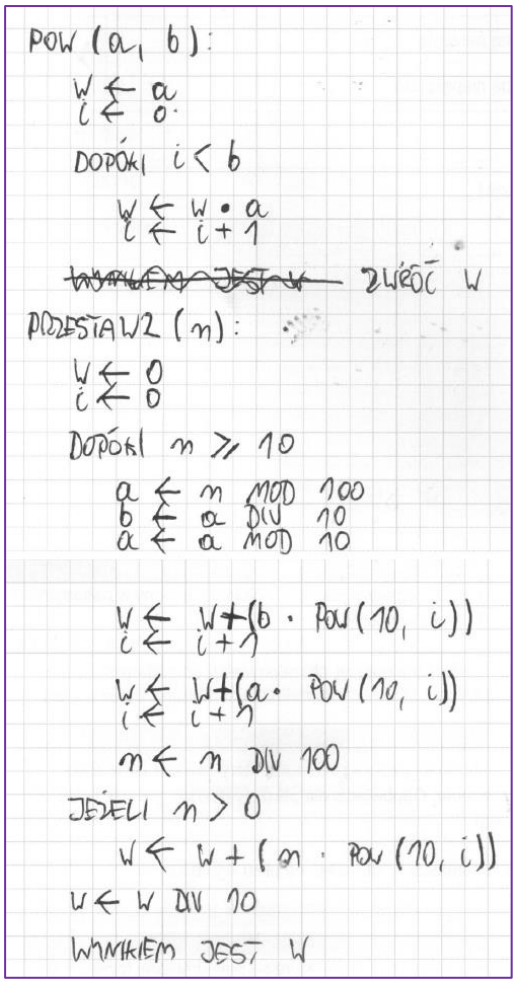
Rysunek 3. Rozwiązanie 2, c++

Warto zauważyć ciekawe podejście zdającego do rozwiązania zadania przedstawionego na Rysunku 3. Zdający do zmiennej *w*, oznaczającej wynik, podstawił najpierw wartość liczby *n*, a w kolejnych krokach odejmował od niej liczby reprezentujące kolejne pary cyfr i dodawał w ich miejsce liczby reprezentujące pary z zamienionymi cyframi.



```
def przestaw2(n):
    k = 1
    w = 0
    while n > 0:
        r = n % 100
        a = r // 10
        b = r % 10
        n = n // 100
        if n > 0:
            w += a * k + b * (k * 10)
        else:
            if a > 0: w += a * k + b * (k * 10)
            else: w += b * k
        k *= 100
    return w
```

Rysunek 4. Rozwiązanie 3, Python



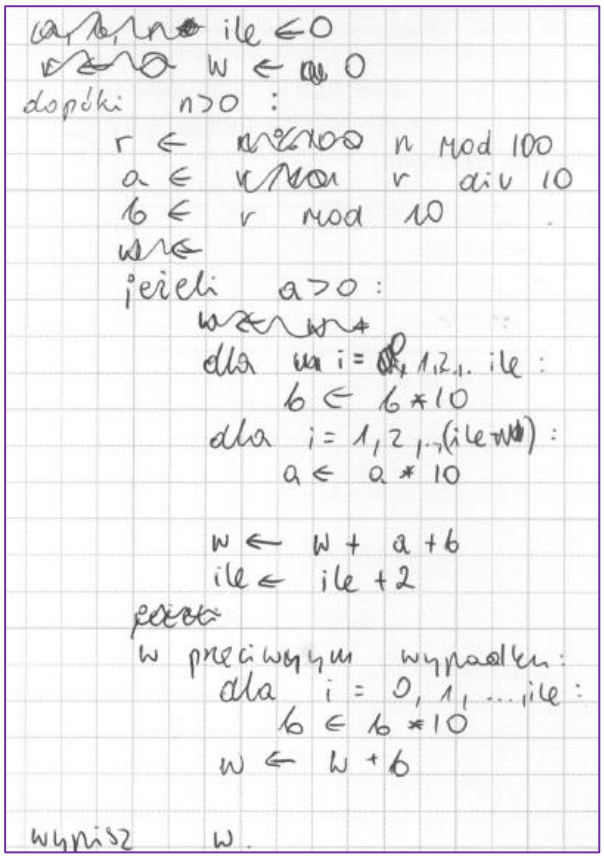
```
POW(a, b):
    W ← a
    i ← 0
    DOPÓKI i < b
        W ← W * a
        i ← i + 1
    ZWRÓĆ W

PRZESTAW2(n):
    W ← 0
    i ← 0
    DOPÓKI n >= 10
        a ← n MOD 100
        b ← a DIV 10
        a ← a MOD 10
        W ← W + (b * POW(10, i))
        i ← i + 1
        W ← W + (a * POW(10, i))
        i ← i + 1
        n ← n DIV 100
    JEŻELI n > 0
        W ← W + (n * POW(10, i))
    W ← W DIV 10
    WYNIKIEM JEST W
```

Rysunek 5. Rozwiązanie w pseudokodzie

Na Rysunku 5. zaprezentowano inne podejście do rozwiązania zadania. Tutaj zdający zapisał najpierw swoją wersję funkcji obliczającej potęgę. Funkcja *POW* dla podanych parametrów *a* i *b* oblicza wartość a^{b+1}. Swoją funkcję zdający poprawnie zastosował w realizacji algorytmu liczącego nierekurencyjnie wartość funkcji przestaw(*n*) – a więc zapewne celowo zapisał ją w ten, a nie inny sposób.

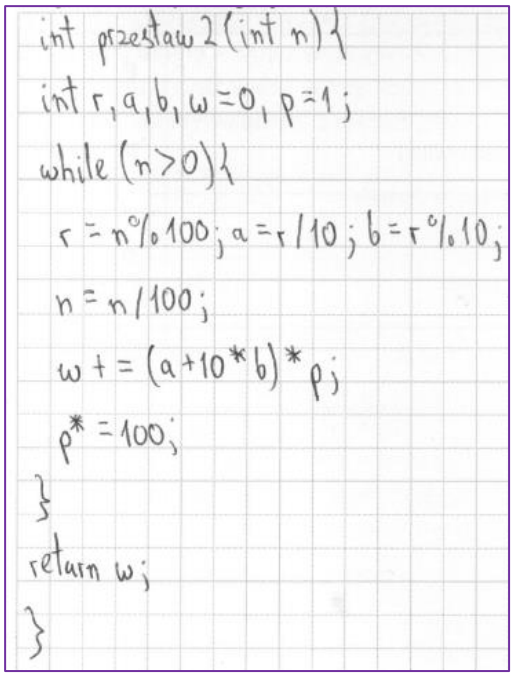
Najczęstsze błędy popełniane w tym zadaniu to:
- nieuwzględnianie nieparzystej liczby cyfr liczby *n* (prawidłowe wyniki dla parzystej liczby cyfr i nieprawidłowe dla nieparzystej)
- problemy z zapisem pętli przechodzącej „po zapisie dziesiętnym" liczby *n* co dwie cyfry
- używanie instrukcji oraz podejścia niezgodnego z wymaganiami zadania zapisanymi w uwadze do zadania.

Przykładowe **błędne** rozwiązania przedstawiono na Rysunkach 6.–8.:



Rysunek 6. Rozwiązanie z błędami, pseudokod

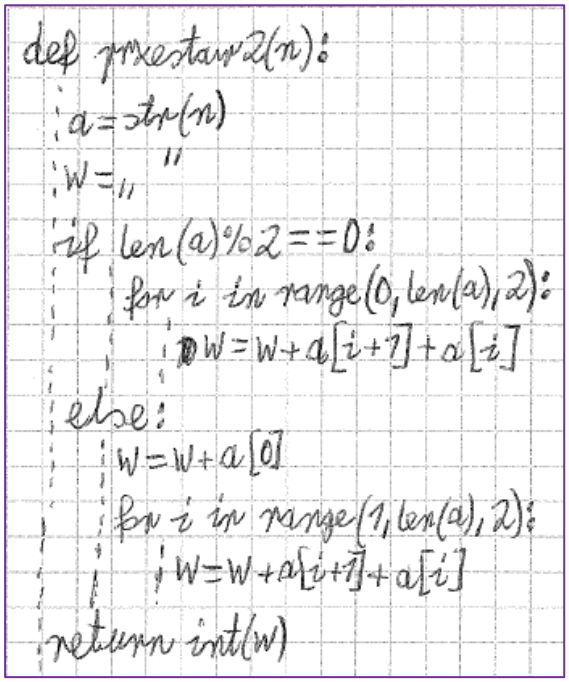
W rozwiązaniu przedstawionym na Rysunku 6. popełniono dwa błędy, które zaowocowały algorytmem działającym w nieskończoność i nie dającym poprawnych wyników. Pierwszy i najbardziej podstawowy, to brak instrukcji zmieniającej wartość liczby *n* po pobraniu z niej do zmiennych *a* i *b* ostatnich dwóch cyfr (*n* = *n* / 100), co spowodowało, że pętla *dopóki* nigdy się nie kończy. Drugim błędem jest o jeden krok za długa pętla zastosowana pod koniec algorytmu (po „w przeciwnym przypadku").



```
int przestaw2(int n){
int r,a,b,w=0,p=1;
while(n>0){
 r=n%100; a=r/10; b=r%10;
 n=n/100;
 w+=(a+10*b)*p;
 p*=100;
}
return w;
}
```

Rysunek 7. Rozwiązanie z błędem, c++

W przypadku rozwiązania przedstawionego na Rysunku 7. zabrakło sprawdzenia, czy w zapisie liczby *n* nie występuje nieparzysta liczba cyfr. Nie sprawdzono, czy w pewnym momencie, po wykonaniu operacji n = n / 100, nie otrzymano liczby mniejszej od 10. W efekcie funkcja działa poprawnie jedynie dla liczb mających w zapisie dziesiętnym parzystą liczbę cyfr, a dla liczb mających nieparzystą liczbę cyfr daje wynik o liczbie cyfr o jeden za dużą i z zerem w miejscu drugiej cyfry, po najbardziej znaczącej. To jest np. dla *n* = 1234 wynik jest poprawny (2143), natomiast dla *n* = 12345 wynikiem jest 103254 zamiast 13254.



```
def przestaw2(n):
    a=str(n)
    w=""
    if len(a)%2==0:
        for i in range(0,len(a),2):
            w=w+a[i+1]+a[i]
    else:
        w=w+a[0]
        for i in range(1,len(a),2):
            w=w+a[i+1]+a[i]
    return int(w)
```

Rysunek 8. Błędne rozwiązanie, Python

Algorytm przedstawiony na Rysunku 8. to rozwiązanie, które nie spełnia warunków zadania. Rozwiązanie w całości oparte jest na operacjach wykonywanych na napisach (w uwadze do zadania wprost jest napisane, że zapisany algorytm może działać wyłącznie na liczbach całkowitych) i używaniu niedozwolonych funkcji wbudowanych (np. *str*, *len*). Takie rozwiązanie nie może zostać uznane za poprawne, ani nawet częściowo poprawne, w tym zadaniu (mimo, że działa w Python).

Tu warto jeszcze raz zwrócić uwagę na zapis przy zadaniu:

Uwaga: Twój algorytm może używać **wyłącznie zmiennych przechowujących liczby całkowite** oraz może operować **wyłącznie na liczbach całkowitych**. W zapisie możesz wykorzystać tylko operacje arytmetyczne: dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, resztę z dzielenia oraz porównywanie liczb, instrukcje sterujące, przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje. **Zabronione** jest używanie funkcji wbudowanych oraz operatorów innych niż wymienione, **w tym – funkcji *przestaw***.

Rysunek 9. Uwaga do zadania 1.3.

Zdający nie zawsze stosują się do wymienionych wymagań, nie zawsze też do końca je rozumieją. Przeanalizujmy więc je dokładniej:

- zapis „Twój algorytm może używać wyłącznie zmiennych przechowujących liczby całkowite oraz może operować wyłącznie na liczbach całkowitych” wyklucza zastosowanie zmiennych innego typu niż całkowitoliczbowy (napisów, typów logicznych, liczb zmiennoprzecinkowych, ale, uwaga: także tablic i list. Kolejne zdania tylko wzmacniają ten zapis

- „W zapisie możesz wykorzystać tylko operacje arytmetyczne […], instrukcje sterujące, przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje” – wolno wykorzystać tylko wymienione operacje, czyli zabronione jest np. potęgowanie.

- „Zabronione jest używanie funkcji wbudowanych oraz operatorów innych niż wymienione […]” – czyli, poza podstawowymi instrukcjami sterującymi (jeżeli, dopóki itp.) i wymienionymi operatorami, nie można użyć żadnych innych, w tym: żadnych konwersji typów, funkcji typu *swap()*, *length()*, *str()*, itp., innych niż wymienione operatorów (np. potęgowania, ale także dodawania znaków zamiast liczb).

Zadanie 2.2. (29% wykonania) i zadanie 2.4. (40% wykonania), to zadania programistyczne należące do jednej wiązki zadań opartej o plik tekstowy zawierający 2000 napisów złożonych z 12 znaków spośród: o, +, *.
W zadaniu 2.2. należało w pliku znaleźć układ znaków tworzący wypełniony kwadrat złożony z 9 jednakowych symboli. Było to drugie co do trudności zadanie w arkuszu egzaminacyjnym. Pierwsza trudność polegała na umiejętnym posługiwaniu się tablicą dwuwymiarową, albo tablicą zawierającą napisy, druga – na przemyśleniu podejścia uwzględniającego wszystkie przypadki (np. położenie symbolu należącego do boku szukanego kwadratu na końcu napisu, na początku napisu albo w pierwszym lub ostatnim wierszu). Wielu zdających nie podjęło próby rozwiązania tego zadania.

W zadaniu 2.2. (Rysunek 10.) należało w pliku znaleźć układ znaków tworzący wypełniony kwadrat, złożony z 9 jednakowych symboli.

Zadanie 2.2. (0–4)

W pliku `symbole.txt` szukamy „kwadratów” złożonych z dziewięciu sąsiadujących identycznych symboli:

```
+ + +        o o o        * * *
+ + +  lub   o o o  lub   * * *
+ + +        o o o        * * *
```

Podaj, ile takich kwadratów występuje w pliku `symbole.txt`. Jeżeli w pliku występuje jeden taki kwadrat, podaj numer wiersza i numer pozycji w wierszu (licząc od 1) jego **środkowego pola**. Jeżeli jest więcej takich kwadratów, podaj numer wiersza i numer pozycji w wierszu dla środkowego pola każdego z nich.

Rysunek 80. Zadanie 2.2.

Było to drugie najtrudniejsze, biorąc pod uwagę poziom wykonania, zadanie w arkuszu egzaminacyjnym. Rozwiązujący to zadanie zdający musieli wykazać się umiejętnością wykorzystania tablicy dwuwymiarowej albo tablicy zawierającej napisy. Zapisanie algorytmu wymagało przemyślanego podejścia, które uwzględni różne przypadki, np. położenie symbolu należącego do boku szukanego kwadratu na końcu napisu, na początku napisu albo w pierwszym lub ostatnim wierszu. Brak tych umiejętności powodował, że wielu zdających nie podjęło próby rozwiązania tego zadania.

Fragment przykładowego poprawnego rozwiązania:

```
for(int i = 0; i < ROZMIAR; i ++)
{
    plik>>wiersz[i];
}

int licznik = 0;

for(int i = 1; i < ROZMIAR - 1; i ++)
{

    for(int j = 1; j < 12; j ++)
    {
        if(wiersz[i][j] == wiersz[i - 1][j - 1]
           and wiersz[i][j] == wiersz[i - 1][j]
           and wiersz[i][j] == wiersz[i - 1][j + 1]
           and wiersz[i][j] == wiersz[i][j - 1]
           and wiersz[i][j] == wiersz[i][j]
           and wiersz[i][j] == wiersz[i][j + 1]
           and wiersz[i][j] == wiersz[i + 1][j - 1]
           and wiersz[i][j] == wiersz[i + 1][j]
           and wiersz[i][j] == wiersz[i + 1][j + 1])
        {
            licznik ++;
            cout<<"wiersz "<<i + 1<<" pozycja "<<j + 1<<endl;
        }
    }

}
cout<<licznik<<endl;
```

Rysunek 11. Fragment rozwiązania zadania 2.2., c++

Zadanie 2.4. było zadaniem należącym do tej samej wiązki. Rozwiązanie tego zadania wymagało umiejętności przeliczania liczb między systemami pozycyjnymi i wykonywania operacji na liczbach zapisanych w systemie trójkowym. Utrudnieniem był fakt, że zamiast liczbami należało posługiwać się odpowiednimi znakami oznaczającymi cyfry 0, 1 i 2.

Zadanie 2.4. (0–3)

Oblicz sumę wszystkich liczb z pliku `symbole.txt`. Podaj jej wartość w zapisie dziesiętnym oraz w zapisie trójkowym z użyciem symboli: `o`, `+`, `*`.

Odpowiedź dla pliku `symbole_przyklad.txt` to
4841542 `+oooo****+oo+o+`

Rysunek 12. Zadanie 2.4.

Poniżej przedstawiono przykładowe poprawne rozwiązanie zdającego, zrealizowane w kilku etapach.
Etap 1 (Rysunek 13.) to: zapisanie funkcji *znak* – mającej zamieniać pojedyncze symbole o, +, * na znaki cyfr 0, 1, 2; zapisanie funkcji *trojkowy* zmieniającej cały napis zapisany za pomocą symboli o, +, * na napis zapisany z użyciem znaków cyfr 0, 1, 2; zapisanie funkcji *dzies* obliczającej wartość dziesiętną liczby podanej w systemie trójkowym w postaci napisu (string).

```cpp
char znak(char x)
{
    if(x == 'o') return '0';
    if(x == '+') return '1';
    if(x == '*') return '2';
}
string trojkowy(string n)
{
    string wynik;
    for(int i = 0; i < n.size(); i ++)
    {
        wynik = wynik + znak(n[i]);
    }
    return wynik;
}
int dzies(string n)
{
    int wynik = 0;
    for(int i = 0; i < n.size(); i ++)
    {
        wynik *= 3;
        wynik += (n[i] - '0');
//        cout<<wynik<<endl;
    }
    return wynik;
}
```

Rysunek 13. Przykładowe rozwiązanie zadania 2.4. cz.1. funkcje pomocnicze. c++

Kolejny etap (Rysunek 14.) to: zapisanie funkcji *na_troj*, która dla podanej liczby w zapisie dziesiętnym zwraca jej zapis w systemie trójkowym w postaci napisu; zapisanie funkcji *symbol* zamieniającej pojedyncze znaki cyfr 0, 1, 2 na symbole o, +, *; zapisanie funkcji *na_symbol* zamieniającej znaki 0, 1, 2 na odpowiednie symbole w całym napisie.

```cpp
string na_troj(int n)
{
    int a;
    string wynik = "";
    while (n > 0)
    {
        a = n % 3;
        n /= 3;
        char znak = '0' + a;
        wynik = znak + wynik;
    }
    return wynik;
}
char symbol(char x)
{
    if(x == '0') return 'o';
    if(x == '1') return '+';
    if(x == '2') return '*';
}
string na_symbol(string n)
{
    string wynik;
    for(int i = 0; i < n.size(); i ++)
    {
        char z = symbol(n[i]);
        wynik = wynik + z;
    }
    return wynik;
}
```

Rysunek 14. Rozwiązanie zadania 2.4. cz.2 - kolejne funkcje pomocnicze. c++

Ostatni etap to zastosowanie zapisanych w poprzednich etapach funkcji w programie (Rysunek 15.)

```
int main()
{
    ifstream plik;
    ofstream odpowiedz;
    plik.open(NAZWA);
    odpowiedz.open("odp.txt");

    string wiersz[ROZMIAR];
    string maxt;
    int maxi = 0;
    int suma = 0;

    for(int i = 0; i < ROZMIAR; i ++)
    {
        plik>>wiersz[i];
        int liczba = dzies(trojkowy(wiersz[i]));
        suma += liczba;
    }

    cout<<suma<<" "<<na_symbol(na_troj(suma));
```

Rysunek 15. Zadanie 2.4. dokończenie rozwiązania. c++

W pełni poprawne rozwiązanie tego zadania można było zapisać również dużo krócej. Jednak omówione powyżej rozwiązanie stanowi bardzo przejrzysty przykład realizacji poszczególnych etapów.

Zadanie 3.2. (33% wykonania) także było zadaniem wymagającym zapisania programu. Jest to zadanie należące do wiązki nazwanej „Dron”. Zadanie oparte było na pliku zawierającym pary liczb oznaczających zmiany położenia w poziomie i pionie pewnego drona. W zadaniu tym należało najpierw wykorzystać podane dane do obliczenia współrzędnych kolejnych punktów położenia drona przy założeniu, że punktem początkowym był punkt o współrzędnych (0,0). Następnie należało operować na otrzymanych współrzędnych.

Zadanie 3.2. (0–4)

Rozważmy **wszystkie punkty**, w których dron znajdował się **po wykonaniu** kolejnych ruchów (przesunięć).

Dla danych z przykładu 1. będą to punkty: (3000, 2000), (5000, 11000), (10000, 4000), (15000, 8000), (18000, 14000) i (20000, 0).

a) Podaj, ile spośród wszystkich rozważanych punktów znajduje się wewnątrz kwadratu o wierzchołkach (0, 0), (0, 5000), (5000, 5000), (5000, 0). Nie liczymy punktów leżących na krawędziach kwadratu.

b) Spośród wszystkich rozważanych punktów znajdź i podaj trzy różne, takie, że jeden z nich jest środkiem odcinka o końcach w pozostałych dwóch. Jest tylko jedna taka trójka punktów.
Uwaga: punkty należące do szukanej trójki nie muszą być trzema kolejnymi punktami, do których przemieszczał się dron.

Rysunek 16. Zadanie 3.2.

Zadanie składało się z dwóch podpunktów. Podpunkt a) sprowadzał się do sprawdzenia, ile z otrzymanych punktów ma tę własność, że jednocześnie pierwsza i druga współrzędna ma wartość dodatnią mniejszą niż 5000. Podpunkt b) okazał się trudniejszy – przede wszystkim należało zauważyć, że współrzędne środka odcinka AB są równe sumie odpowiednich współrzędnych punktów A i B podzielonej przez 2. Poprawną odpowiedź do tego podpunktu można było uzyskać na kila sposobów. Jednym z nich jest sprawdzenie opisanego powyżej warunku na wszystkich możliwych trójkach punktów (z uwzględnieniem, że tę samą trójkę punktów otrzymamy dwa razy, w zależności od ustalenia początku i końca odcinka). Innym sposobem jest rozważanie punktów w kolejności ich występowania w pliku (tj. zawsze pierwszy punkt z pierwszą współrzędną mniejszą niż pierwsza współrzędna drugiego punktu i trzeci punkt z pierwszą współrzędną większą niż pierwsza współrzędna drugiego punktu), pamiętając nadal, że nie muszą to być trzy kolejne punkty.

Przykładowe rozwiązanie przedstawiają rysunki 17. i 18.:

```
void zad2a()
{
    ifstream plik;
    plik.open("dron.txt");
    int k=0;
    int x=0;
    int y=0;
    for (int i=0; i<100; i++){
        int a;
        int b;
        plik >> a;
        plik >> b;
        x = x + a;
        y = y + b;
        if ((x>0 && x<5000) && (y>0 && y<5000)){
            k++;
        }
    }
    cout << k;
}
```

Rysunek 17. Rozwiązanie zadania 3.2. a). c++

```
void zad2b()
{
    ifstream plik;
    plik.open("dron.txt");
    int x[100] = {0};
    int y[100] = {0};
    plik >> x[0];
    plik >> y[0];
    for (int i=1; i<100; i++){
        int a;
        int b;
        plik >> a;
        plik >> b;
        x[i] = x[i-1] + a;
        y[i] = y[i-1] + b;
    }
    for (int i=0; i<100-2; i++){
            for (int j=i+2; j<100; j++){
                double xs = (x[i] + x[j])/2;
                double ys = (y[i] + y[j])/2;
                for (int p=0; p<100; p++){
                    if (x[p]==xs && y[p]==ys){
                        cout << "(" << x[i] << ", " << y[i]
                        << "), " << "(" << xs << ", " << ys << "), " << "(" << x[j] << ", " << y[j] << ")";
                    }
                }
            }
    }
}
```

Rysunek 18. Rozwiązanie zadania 3.2. b). c++

Zadanie 7.4. (31%) oraz Zadanie 7.3. (35%), to zadania z ostatniej wiązki zadań w arkuszu – wiązki bazodanowej „Poszukiwanie wody na Marsie”. Zadanie oparte zostało na danych dotyczących hipotetycznej eksploracji Marsa w poszukiwaniu wody z pomocą łazików.

Zadanie 7. Poszukiwanie wody na Marsie

W trzech plikach tekstowych o nazwach `laziki.txt`, `obszary.txt`, `pomiary.txt` zapisano informacje zawierające dane o poszukiwaniu wody na Marsie w latach 2050–2080. Łaziki zasilane energią słoneczną poruszają się po różnych obszarach Marsa i wykonują pomiary georadarowe, na podstawie których szacują ilość wody i głębokość, na której się ona znajduje. Pierwszy wiersz każdego z plików jest wierszem nagłówkowym, a dane w wierszach rozdzielono znakami tabulacji.

Plik o nazwie `laziki.txt` zawiera informacje o różnych łazikach, które wykonywały pomiary. W każdym wierszu tego pliku znajdują się:

`nr_lazika` – co najwyżej trzycyfrowy, unikatowy numer łazika
`nazwa_lazika` – nazwa łazika (tekst do 50 znaków)
`rok_wyslania` – rok startu z Ziemi
`wsp_ladowania` – współrzędne lądowania na Marsie oddzielone znakiem przecinka i spacją

Przykład:

```
nr_lazika nazwa_lazika   rok_wyslania   wsp_ladowania
1         Mariner 3      2049           50.51N, 70.01E
2         Mariner 6      2050           11.90N, 119.49E
3         Mariner 7      2050           44.90S, 130.80W
```

Plik o nazwie `obszary.txt` zawiera informacje o obszarach na Marsie. W każdym wierszu tego pliku znajdują się:

`kod_obszaru` – pięcioznakowy, unikatowy kod obszaru
`nazwa_obszaru` – nazwa obszaru (tekst do 50 znaków)

Przykład:

```
kod_obszaru    nazwa_obszaru
MC-01          Mare Boreum
MC-02          Diacria
MC-03          Arcadia
```

Plik o nazwie `pomiary.txt` zawiera informacje o wynikach badań georadarowych wykonanych przez łaziki. W każdym wierszu tego pliku znajdują się:

`nr_lazika` – co najwyżej trzycyfrowy numer łazika
`data_pomiaru` – data wykonania pomiaru (w formacie rrrr-mm-dd)
`kod_obszaru` – pięcioznakowy kod obszaru, na którym został wykonany pomiar
`wspolrzedne` – współrzędne wykonania pomiaru, oddzielone znakiem przecinka i spacją
`glebokosc` – szacowana głębokość, na której znajduje się woda (w metrach)
`ilosc` – szacowana ilość wody (w m^3)

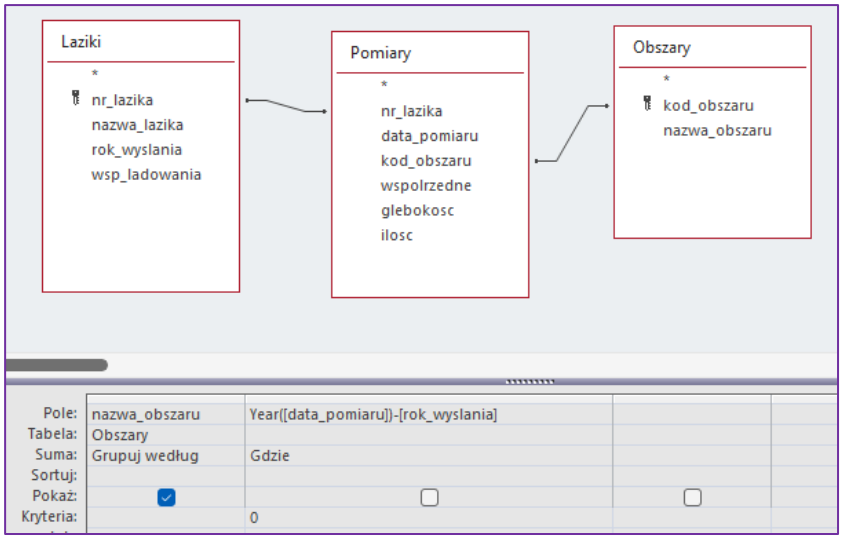
Rysunek 19. Zadanie 7 - Poszukiwanie wody na Marsie.

Zadanie 7.3. (0–2)
Podaj nazwy obszarów na Marsie, na których żaden z łazików nie wykonał ani jednego pomiaru w tym samym roku, w którym został wysłany z Ziemi.

Rysunek 20. Treść zadania 7.3.

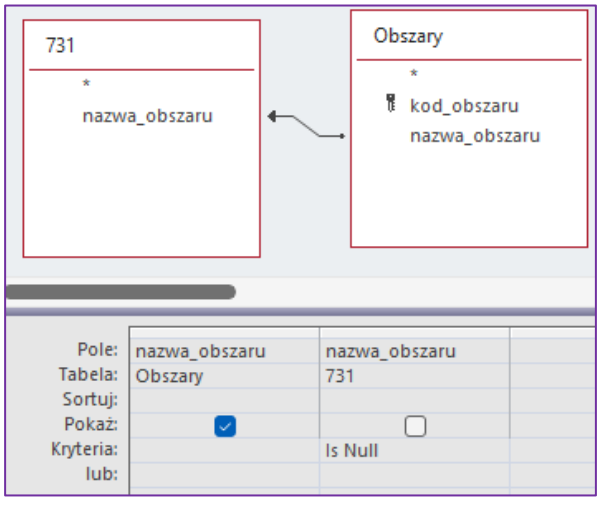
Zadanie 7.3. było typowym zadaniem polegającym na wyszukiwaniu niepasujących danych. Aby znaleźć obszary, na których żaden z łazików nie wykonywał pomiarów w roku swojego wysłania z Ziemi, najpierw należało znaleźć takie obszary, na których taka sytuacja się zdarzyła. A więc pierwszym etapem zadania było utworzenie kwerendy dającej w wyniku nazwy obszarów, na których łaziki wykonywały pomiary w roku swojego wysłania z Ziemi.

Poniżej (Rysunek 21.) przedstawione zostało przykładowe rozwiązanie zdającego, stanowiące pierwszy etap wykonania zadania 7.3. W kwerendzie zastosowano funkcję *Year* (MSAccess) dającą w wyniku rok pobrany z daty – w tym przypadku z pola *data_pomiaru*.



Rysunek 21. Rozwiązanie zadania 7.3. 1 etap.

Następnym etapem jest wykonanie kwerendy wybierającej spośród wszystkich nazw obszarów, nazwy tych, które nie występują w pierwszej kwerendzie (Rysunek 22.).



Rysunek 22. Kwerenda dająca wyniki zadania 7.3.

Zapytanie SQL odpowiadające kwerendzie przedstawionej na Rysunku 13., w której „731” jest nazwą poprzedniej kwerendy, to:

```
SELECT Obszary.nazwa_obszaru
FROM Obszary LEFT JOIN 731 ON Obszary.nazwa_obszaru = [731].nazwa_obszaru
WHERE ((([731].nazwa_obszaru) Is Null));
```

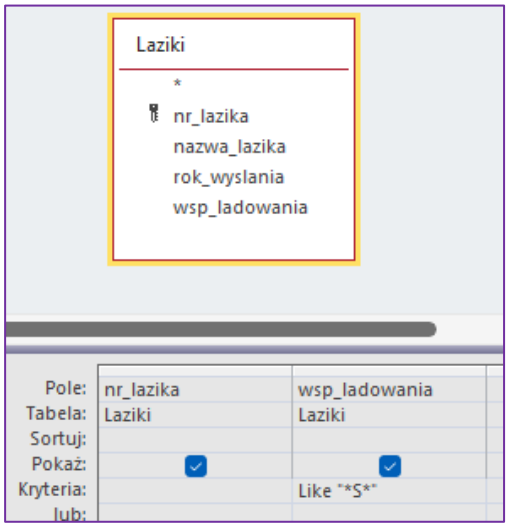
Wynik:
Amazonis
Arabia
Syrtis Major
Elysium
Sinus Sabaeus
Mare Tyrrhenum
Aeolis
Eridania

Zadanie 7.4. (0–2)
Podaj nazwy łazików, które wylądowały na półkuli południowej, ale wykonywały pomiary na obu półkulach: północnej (N) i południowej (S).

Rysunek 23. Treść zadania 7.4.

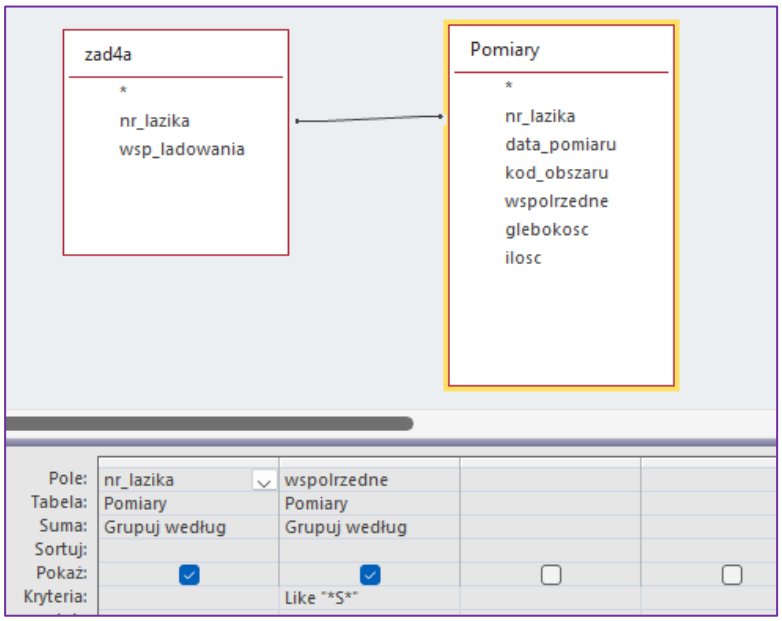
Zadanie 7.4. (Rysunek 23.) okazało się trzecim najtrudniejszym dla zdających zadaniem w arkuszu. Jego rozwiązanie można było otrzymać w kilku dość prostych krokach. Dane dotyczące współrzędnych lądowania, jak i wykonania pomiarów przez łaziki, można było importować jako zwykły tekst. Zatem sprawdzenie, na której półkuli łazik wylądował i na której półkuli wykonywano pomiar sprowadzało się do sprawdzenia, czy współrzędne zawierają znak S czy N. Poniżej przedstawiono jeden ze sposobów rozwiązania wybrany przez zdającego:

Etap 1 (Rysunek 24.): wyszukanie numerów łazików, które wylądowały na półkuli południowej.



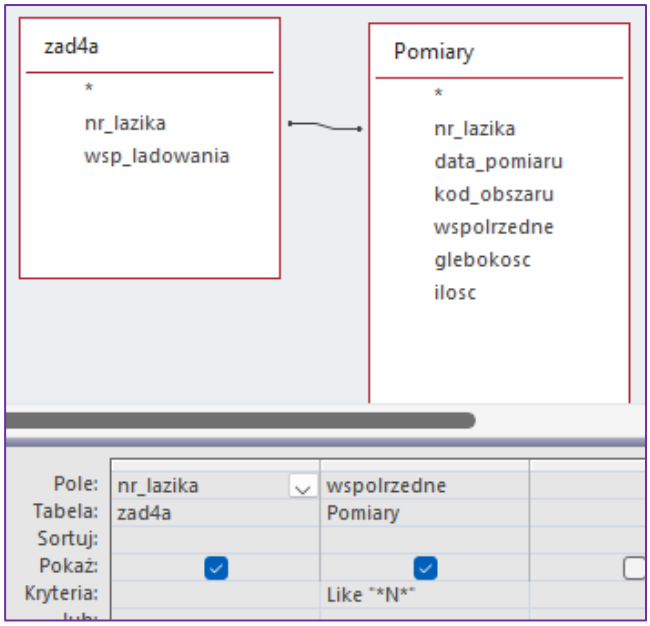
Rysunek 249. Rozwiązanie zadania 7.4. etap 1 - łaziki, które wylądowały na półkuli południowej.

Etap 2 (Rysunek 25.): wyszukanie numerów łazików, które wylądowały na półkuli południowej i wykonywały pomiary na półkuli południowej.



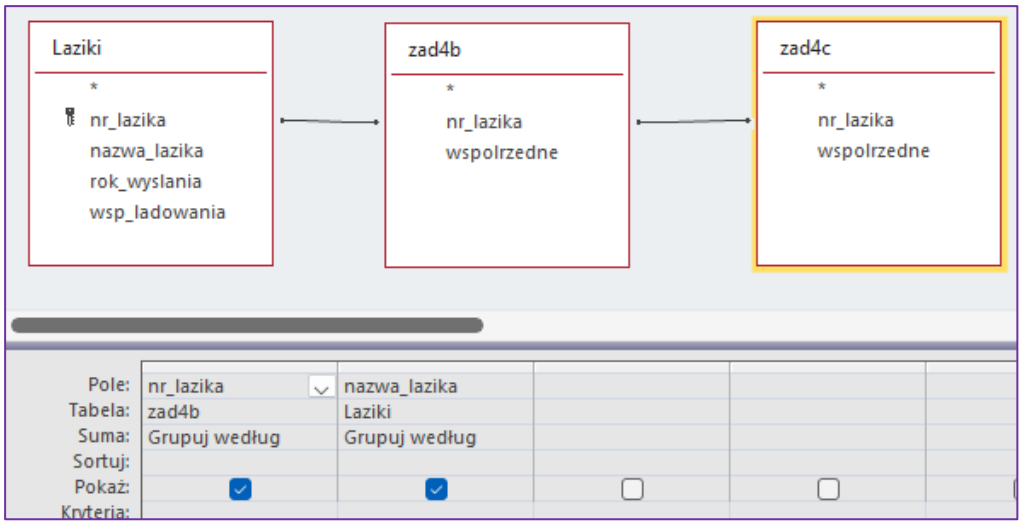
Rysunek 25. Rozwiązanie zadania 7.4. etap 2 - łaziki, które wykonywały pomiary na półkuli południowej.

Etap 3 (Rysunek 26.): wyszukanie numerów łazików, które wylądowały na półkuli południowej i wykonywały pomiary na półkuli północnej.



Rysunek 26. Rozwiązanie zadania 7.4. etap 3 - łaziki, które wykonywały pomiary na półkuli północnej.

Mając wszystkie trzy zbiory numerów łazików należało w ostatniej kwerendzie wybrać nazwy tych, które spełniały wszystkie 3 warunki (Rysunek 27.).



Rysunek 27. Zadanie 7.4. kwerenda końcowa.

```
SELECT zad4b.nr_lazika, Laziki.nazwa_lazika
FROM (zad4c INNER JOIN zad4b ON zad4c.nr_lazika = zad4b.nr_lazika) INNER JOIN
Laziki ON zad4b.nr_lazika = Laziki.nr_lazika
GROUP BY zad4b.nr_lazika, Laziki.nazwa_lazika;
```

Wynik:
Mariner 14
Mariner 15
Mariner 20
Viking 17
Spirit 7
Spirit 12
Rosetta 1
Rosetta 8
Phoenix 3
Phoenix 13

Zadania, z którymi zdający poradzili sobie najlepiej

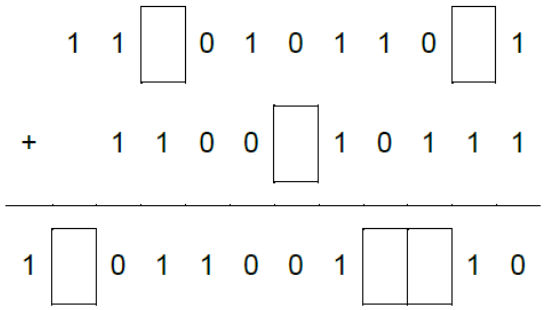
W tej części komentarza omówimy zadania, z którymi zdający poradzili sobie najlepiej. Przyjmiemy do analizy zadania, których poziom wykonania jest wyższy niż 60%. Takich zadań w arkuszu jest 5.

Zadania, których poziom wykonania jest wyższy niż 60%, wymieniając kolejno od najłatwiejszego, to (w nawiasach oprócz poziomu wykonania zapisano jakiej tematyki dotyczy zadanie oraz rodzaj zadania):
1. Zadanie 5. (80%, zadanie z zakresu algorytmiki – dodawanie liczb binarnych, otwarte),
2. Zadanie 4. (73%, zadanie teoretyczne, zamknięte),
3. Zadanie 6.2. (72%, zadanie dotyczące arkusza kalkulacyjnego, praktyczne),
4. Zadanie 1.2. (72%, zadanie z zakresu algorytmiki – analiza algorytmu, otwarte),
5. Zadanie 1.1. (66%, zadanie z zakresu algorytmiki – analiza algorytmu, otwarte).

Zadanie 5. (Rysunek 28.) było dla zdających najłatwiejszym zadaniem w arkuszu. W tym zadaniu należało uzupełnić brakujące cyfry w zapisie dodawania dwóch liczb binarnych.

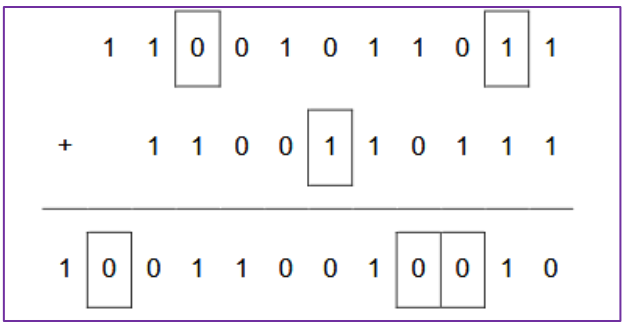
Zadanie 5. (0–2)

Poniżej sposobem pisemnym dodano dwie liczby podane w zapisie binarnym. Uzupełnij brakujące cyfry tak, aby działanie było wykonane poprawnie.



Rysunek 28. Zadanie 5

Rozwiązanie:



Rysunek 29. Rozwiązanie zadania 5.

Zadanie 4. (Rysunek 30.) było zadaniem zamkniętym wymagającym znajomości podstawowych zagrożeń komputerowych.

Zadanie 4. (0–1)

Dokończ zdanie. Zaznacz właściwą odpowiedź spośród podanych.

Program typu *keylogger* służy do

A. szyfrowania informacji do postaci uniemożliwiającej jej odczytanie bez zdefiniowanego klucza.
B. przechowywania danych logowania, w tym haseł, w bezpiecznym miejscu na dysku użytkownika.
C. generowania kodu, który umożliwia użytkownikowi bankowości elektronicznej wykonanie operacji.
D. przechwytywania i gromadzenia informacji o naciśniętych klawiszach.

Rysunek 30. Zadanie 4.

Prawidłowa odpowiedź to D.

Zadanie 6.2. (Rysunek 31.) to zadanie praktyczne z wiązki zadań dotyczącej arkusza kalkulacyjnego o nazwie „Martianeum”. Najszybszym sposobem rozwiązania tego zadania jest zastosowanie tabeli przestawnej.

Zadanie 6.2. (0−1)

Podaj nazwę obszaru, dla którego średnia masa przywiezionych ładunków jest najmniejsza.

Rysunek 31. Zadanie 6.2.

Na Rysunku 32. przedstawiono przykładowe rozwiązanie zadania:

	A	B	C	D	E
1					
2					
3	**Etykiety wierszy**	**Średnia z masa [kg]**			
4	*Thaumasia*	*17,83571429*			
5	Amenthes	18,39821429			
6	Argyre	18,4516129			*ZADANIE 6.2*
7	Elysium	18,58717949			*ODP:*
8	Cebrenia	18,83076923			**Thaumasia**
9	Mare Acidalium	19,109375			
10	Hellas	19,11333333			
11	Arcadia	19,15283019			
12	Ismenius Lacus	19,3325			
13	Iapygia	19,36225166			
14	Eridania	19,525			

Rysunek 32. Rozwiązanie zadania 6.2.

Zadania 1.1. (Rysunek 34.) i 1.2. (Rysunek 36.) należą do wiązki o nazwie „Funkcja rekurencyjna”. Na początku wiązki podano zapis pewnej funkcji rekurencyjnej (mod jest operatorem reszty z dzielenia, a div – dzielenia całkowitego):

Zadanie 1. Funkcja rekurencyjna

Dana jest rekurencyjna funkcja *przestaw*, której parametrem jest nieujemna liczba całkowita:

```
przestaw(n):
    r ← n mod 100
    a ← r div 10
    b ← r mod 10
    n ← n div 100
    jeżeli n > 0
        w ← a + 10 * b + 100 * przestaw(n)
    w przeciwnym razie
        jeżeli a > 0
            w ← a + 10 * b
        w przeciwnym razie
            w ← b
    wynikiem jest w
```

Rysunek 33. Zadanie 1.

Zadania 1.1. i 1.2. sprawdzały umiejętność analizy funkcji *przestaw(n)*, której definicję przedstawiono na Rysunku 24. W wyniku działania tej funkcji dochodziło do zamiany miejscami w parach cyfr zapisu dziesiętnego liczby podanej jako parametr, zaczynając od cyfry najmniej znaczącej, tj. ostatniej z przedostatnią, potem trzeciej od końca z czwartą itd. Na przykład dla liczby 1234 wynikiem działania funkcji było 2143. W przypadku liczby o nieparzystej liczbie cyfr w zapisie dziesiętnym na swoim miejscu pozostawała cyfra najbardziej znacząca (pierwsza „od lewej") – np. dla liczby 12345 wynik działania funkcji to 13254.

W zadaniu 1.1. należało podać wyniki działania funkcji dla przykładowych danych i liczby jej wywołań podczas obliczania tych wyników. W przypadku tego zadania zdający mógł przeprowadzić analizę ręcznie lub wykorzystać komputer do zapisania funkcji w postaci programu w wybranym języku programowania i sprawdzenia wyników.

Zadanie 1.1. (0–3)

Uzupełnij tabelę – wpisz w drugiej kolumnie wynik funkcji *przestaw*(*n*) dla podanych wartości argumentu *n* oraz wpisz w trzeciej kolumnie liczbę wywołań funkcji *przestaw* łącznie z pierwszym wywołaniem z parametrem *n*.

n	**Wynik działania funkcji *przestaw***	**Liczba wywołań funkcji *przestaw***
316498	134689	3
43657688		
154005710		
998877665544321		

Rysunek 34. Zadanie 1.1.

Wyniki:

n	**Wynik działania funkcji** *przestaw*	**Liczba wywołań funkcji** *przestaw*
316498	134689	3
43657688	**34566788**	**4**
154005710	**145007501**	**5**
998877665544321	**989786756453412**	**8**

Rysunek 35. Rozwiązanie zadania 1.1.

W zadaniu 1.2. należało określić liczbę wywołań funkcji w zależności od długości zapisu dziesiętnego (liczby cyfr) parametru *n*. Z tym zadaniem także większość zdających nie miała problemów.

Zadanie 1.2. (0–2)

Oceń prawdziwość podanych zdań. Zaznacz **P**, jeśli zdanie jest prawdziwe, albo **F** – jeśli jest fałszywe.

Niech *n* będzie liczbą *k*-cyfrową, gdzie $k > 0$. Liczba wywołań funkcji *przestaw* w zależności od *k* jest równa:

1.	$\frac{k}{2}$	**P**	**F**
2.	$(k+1)\ \mathrm{div}\ 2$ (gdzie div oznacza dzielenie całkowite)	**P**	**F**
3.	$\begin{cases} \frac{k}{2} & \text{gdy } k \text{ jest liczbą parzystą} \\ \frac{k+1}{2} & \text{gdy } k \text{ jest liczbą nieparzystą} \end{cases}$	**P**	**F**
4.	$\frac{k+1}{2}$	**P**	**F**

Rysunek 36. Zadanie 1.2.

Prawidłowa odpowiedź to FPPF.

Wnioski i rekomendacje

1. W tym roku po raz drugi do egzaminu w nowej formule przystępowali absolwenci techników. Liczba tych absolwentów była jednak mniejsza niż w poprzednich latach. Prawdopodobnie część z nich skorzystała z możliwości zamiany konieczności zdawania przedmiotu na poziomie rozszerzonym na wyniki z egzaminu zawodowego. Średni wynik uzyskany przez absolwentów techników – podobnie jak w poprzednich latach – jest niższy niż średni wynik absolwentów liceów.
2. Poziom wykonania poszczególnych zadań wskazuje, że w arkuszu nie było zadań określanych jako bardzo łatwe dla zdających, jak również jako bardzo trudne. Najwięcej trudności sprawiło zdającym, podobnie jak w ubiegłych latach, zadanie wymagające zapisu algorytmu lub programu. W tym roku jednym z trudniejszych okazało się także zadanie bazodanowe.
3. W przypadku zadania wymagającego napisania algorytmu w arkuszu egzaminacyjnym zdający często nie uwzględniali warunków zapisanych w uwadze pod treścią zadania, dlatego podczas zajęć lekcyjnych należy:
   - kłaść nacisk na uważne czytanie i analizowanie treści zadania w całości
   - ćwiczyć umiejętność zapisywania algorytmu, także w przypadku napisania go w pierwszej kolejności przy pomocy komputera, tak aby używać jedynie dozwolonych instrukcji i operacji języka programowania.
4. W przypadku zadań praktycznych należy każdorazowo podkreślać, że zapisanie jedynie odpowiedzi w pliku tekstowym z wynikami, nie stanowi podstawy do przyznania punktów – nawet, jeśli wyniki są prawidłowe. Zapisane odpowiedzi zostaną ocenione tylko wtedy, gdy są odzwierciedleniem komputerowej realizacji rozwiązania zadania. Dlatego należy zwracać uwagę na konieczność czytelnego wskazania odpowiedzi, np.:
   - umieszczenie i wyróżnienie kolorem rozwiązania każdego zadania w odrębnej, odpowiednio nazwanej zakładce arkusza kalkulacyjnego
   - zapisywanie kwerend pod nazwami wskazującymi na numer zadania
   - wpisywanie nazw plików zawierających kody źródłowe programów, itp.
5. Przy rozwiązywaniu wszystkich zadań praktycznych należy również zwracać uwagę na:
   - dobór odpowiednich narzędzi do realizacji rozwiązania zadania – jeśli np. w poleceniu zapisano „napisz program”, to rozwiązanie powinno być zapisane w pliku zawierającym kod źródłowy programu
   - dokładne analizowanie treści zadania, np. typ wykresu, jaki należy wstawić i jego opis
   - planowanie czasu przeznaczonego na rozwiązywanie poszczególnych zadań.