CENTRALNA KOMISJA EGZAMINACYJNA

OKE OKRĘGOWA KOMISJA EGZAMINACYJNA W POZNANIU

Rodzaj dokumentu:	**Sprawozdanie za rok 2023 województwo zachodniopomorskie**
Egzamin:	**Egzamin maturalny**
Przedmiot:	**Informatyka**
Poziom:	**Poziom rozszerzony**
Termin egzaminu:	22 maja 2023 r.
Data publikacji dokumentu:	19 września 2023 r.

Opracowanie
Iwona Arcimowicz (Centralna Komisja Egzaminacyjna)
Ewa Kałucka (Okręgowa Komisja Egzaminacyjna w Jaworznie)

Redakcja
dr Wioletta Kozak (Centralna Komisja Egzaminacyjna)

Opracowanie techniczne
Andrzej Kaptur (Centralna Komisja Egzaminacyjna)

Współpraca
Beata Dobrosielska (Centralna Komisja Egzaminacyjna)
Agata Wiśniewska (Centralna Komisja Egzaminacyjna)
Pracownie ds. Analiz Wyników Egzaminacyjnych okręgowych komisji egzaminacyjnych

Opracowanie dla województwa zachodniopomorskiego

Okręgowa Komisja Egzaminacyjna w Poznaniu
Izabela Szafrańska
Anna Sperling

Centralna Komisja Egzaminacyjna
ul. Józefa Lewartowskiego 6, 00-190 Warszawa
tel. 22 536 65 00, fax 22 536 65 04
e-mail: sekretariat@cke.gov.pl
www.cke.gov.pl

Spis treści

Opis arkusza maturalnego 4
Dane dotyczące populacji zdających 4
Przebieg egzaminu 5
Podstawowe dane statystyczne 6
Komentarz 16
Wnioski i rekomendacje 32

Opis arkusza egzaminu maturalnego

W roku szkolnym 2022/2023 egzamin maturalny z informatyki został przeprowadzany na podstawie wymagań egzaminacyjnych określonych w rozporządzeniu Ministra Edukacji i Nauki z dnia 10 czerwca 2022 r.[1]

Arkusz egzaminacyjny z informatyki na poziomie rozszerzonym zawierał ogółem 23 zadania w tym 21 zadań w 5 wiązkach tematycznych.

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 (6 zadań łącznie za 14 punktów, w tym jedno zadanie zamknięte za 1 punkt).
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 (16 zadań łącznie za 35 punktów, w tym 15 zadań praktycznych – wymagających użycia komputera i zapisania programu, wykorzystania arkusza kalkulacyjnego lub bazy danych).
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 (jedno zadanie z zakresu bezpieczeństwa danych za 1 punkt).

Egzamin trwał 210 minut. Za rozwiązanie wszystkich zadań można było otrzymać 50 punktów.

Dane dotyczące populacji zdających

TABELA 1. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZU STANDARDOWYM*

Liczba zdających (Formuła 2023)		**89**
Zdający rozwiązujący zadania w arkuszu standardowym	ze szkół na wsi	0
	ze szkół w miastach do 20 tys. mieszkańców	3
	ze szkół w miastach od 20 tys. do 100 tys. mieszkańców	32
	ze szkół w miastach powyżej 100 tys. mieszkańców	54
	ze szkół publicznych	78
	ze szkół niepublicznych	11
	kobiety	8
	mężczyźni	81
	bez dysleksji rozwojowej	80
	z dysleksją rozwojową	9
	o których mowa w art. 2 ust. 1 ustawy[2] (obywatele Ukrainy)	1

* Dane w tabeli dotyczą tegorocznych absolwentów.

[1] Rozporządzenie Ministra Edukacji i Nauki z dnia 10 czerwca 2022 r. w sprawie wymagań egzaminacyjnych dla egzaminu maturalnego przeprowadzanego w roku szkolnym 2022/2023 i 2023/2024 (poz. 1246).
[2] Ustawa z dnia 12 marca 2022 r. o pomocy obywatelom Ukrainy w związku z konfliktem zbrojnym na terytorium tego państwa (poz. 583, z późn. zm.).

Z egzaminu w Formule 2023 i Formule 2015 zwolniono 1 osobę – laureata/finalistę Olimpiady Informatycznej.

TABELA 2. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZACH DOSTOSOWANYCH

Zdający rozwiązujący zadania w arkuszach dostosowanych	z autyzmem, w tym z zespołem Aspergera	3
	słabowidzący	0
	niewidomi	0
	słabosłyszący	0
	niesłyszący	1
	z niepełnosprawnością ruchową spowodowaną mózgowym porażeniem dziecięcym	0
	z zaburzeniem widzenia barw	0
	Ogółem	**4**

Przebieg egzaminu

TABELA 3. INFORMACJE DOTYCZĄCE PRZEBIEGU EGZAMINU

Termin egzaminu			22 maja 2023
Czas trwania egzaminu dla arkusza standardowego			210 minut
Liczba szkół			31
Liczba zespołów egzaminatorów			0
Liczba egzaminatorów			0
Liczba obserwatorów[3] (§ 8 ust. 1)			0
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	0
	art. 44zzy ust. 10	niemożność ustalenia wyniku (np. zaginięcie karty odpowiedzi)	0
Liczba wglądów[4] (art. 44zzz)			3

[3] Rozporządzenie Ministra Edukacji i Nauki z dnia 1 sierpnia 2022 r. w sprawie egzaminu maturalnego (poz. 1644) – podano łącznie dla Formuły 2023 i Formuły 2015.

[4] Ustawa z dnia 7 września 1991 r. o systemie oświaty (Dz.U. z 2022 r. poz. 2230).

Podstawowe dane statystyczne

Wyniki zdających

WYKRES 1. ROZKŁAD WYNIKÓW ZDAJĄCYCH

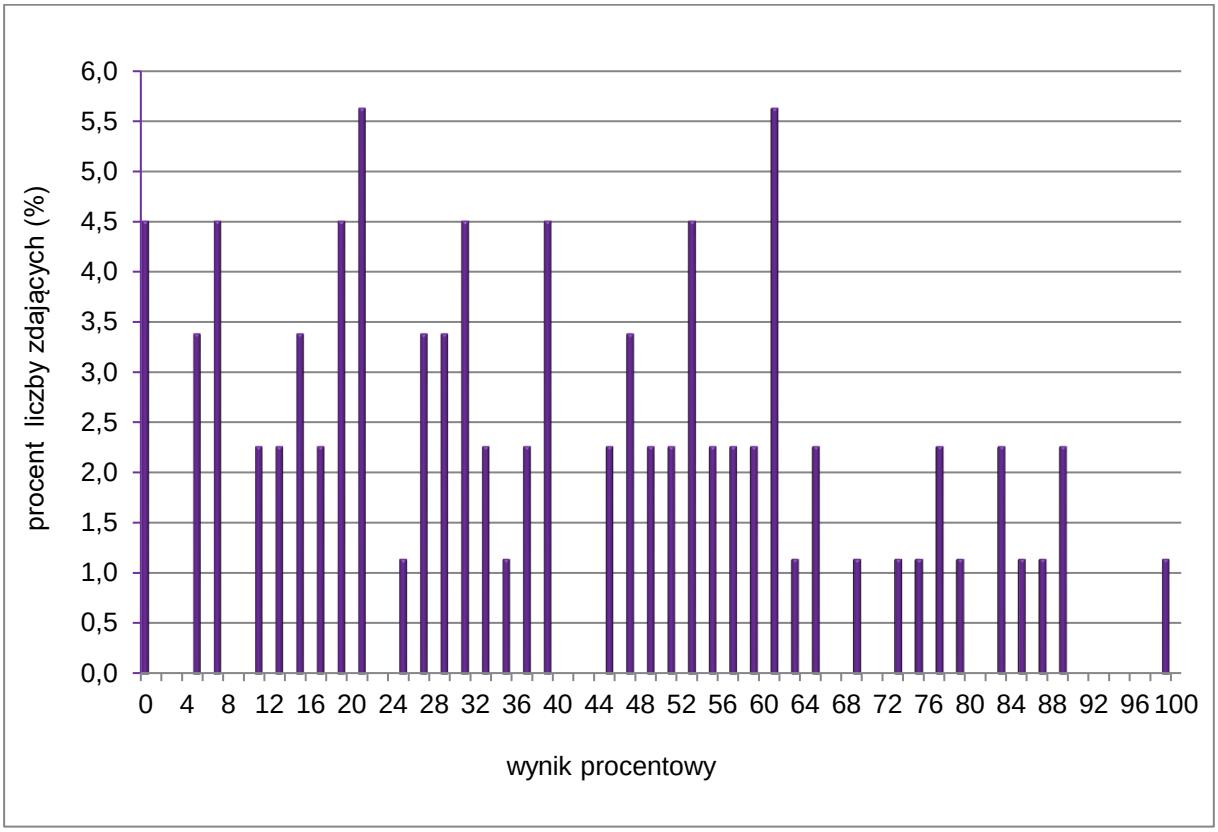


TABELA 4. WYNIKI ZDAJĄCYCH – PARAMETRY STATYSTYCZNE*

Zdający	Liczba zdających	Minimum (%)	Maksimum (%)	Mediana (%)	Modalna (%)	Średnia (%)	Odchylenie standardowe (%)
Ogółem Formuła 2023	**89**	**0**	**100**	**38**	**22-62 (2)**	**41**	**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 egzaminacyjne 2023			
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.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. P.I.3) […] stosuje podejście zachłanne i rekurencję; P.I.5) sprawdza poprawność działania algorytmów dla przykładowych danych.	59
1.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. P.I.3) […] stosuje podejście zachłanne i rekurencję; P.I.5) sprawdza poprawność działania algorytmów dla przykładowych danych.	57
1.3.	I. Rozumienie, analizowanie i rozwiązywanie problemów.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. P.I.3) […] stosuje podejście zachłanne i rekurencję; P.I.5) sprawdza poprawność działania algorytmów dla przykładowych danych.	65
2.1.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych; I.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. I+II. 1) zapisuje za pomocą listy kroków, schematu blokowego 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: a) na liczbach: […] zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi […].	37

2.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych; I.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. II.2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów; II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	49
2.3.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych; I.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. II.2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów; II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	43
2.4.	I. Rozumienie, analizowanie i rozwiązywanie problemów.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych; I.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. 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 […].	39

2.5.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych; I.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb […]. II.2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów; II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...].	29
3.1.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. I+II. 2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: c) znajdowania w ciągu podciągów o różnorodnych własnościach […].	44

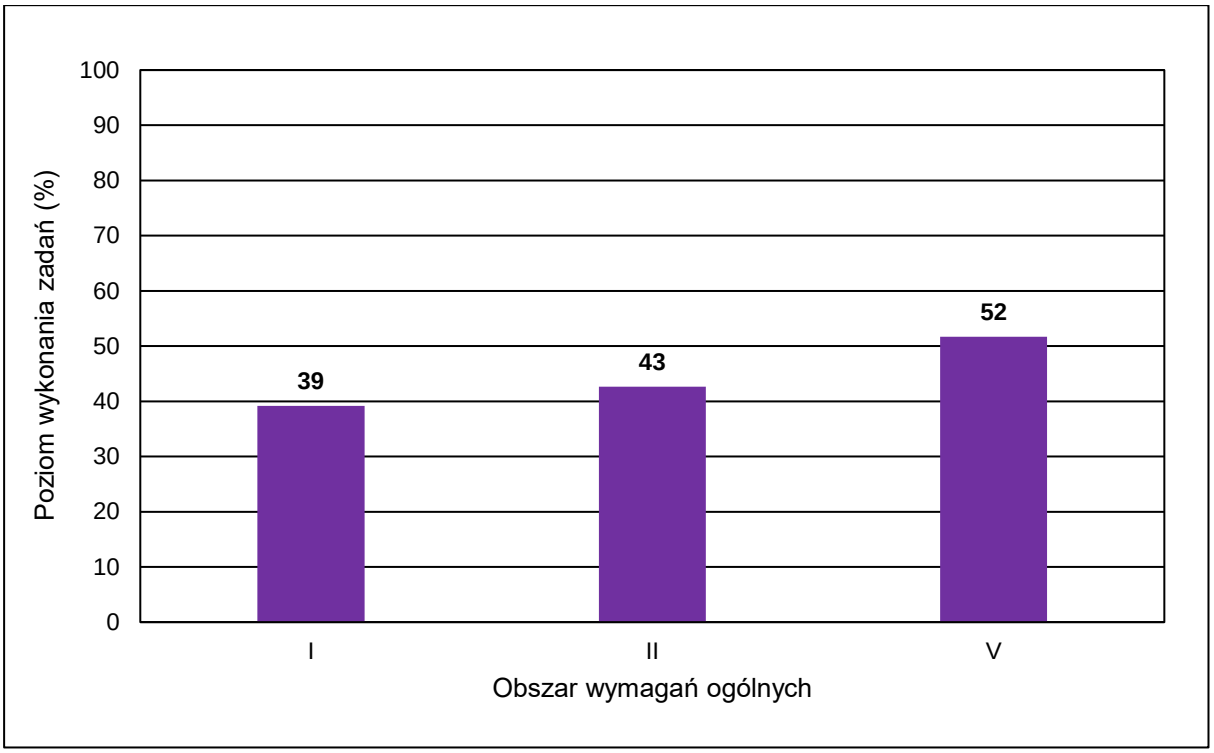
3.2.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. I+II. 2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: c) znajdowania w ciągu podciągów o różnorodnych własnościach […].	36
3.3.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. I+II. 2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: c) znajdowania w ciągu podciągów o różnorodnych własnościach […].	4
3.4.	I. Rozumienie, analizowanie i rozwiązywanie problemów. II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: I.2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. II.3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów. 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, stosuje przy tym: instrukcje	6

		wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. I+II. 2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów: c) znajdowania w ciągu podciągów o różnorodnych własnościach […].	
4.	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.	Zdający: P.V.3) […] objaśnia rolę szyfrowania informacji; V.1) objaśnia rolę technik uwierzytelniania, kryptografii i podpisu elektronicznego w ochronie i dostępie do informacji; V.2) omawia znaczenie algorytmów szyfrowania i składania podpisu elektronicznego.	52
5.	I. Rozumienie, analizowanie i rozwiązywanie problemów.	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.5) przedstawia sposoby reprezentowania w komputerze znaków, liczb, […].	74
6.1.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) 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: b) 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.	70
6.2.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) 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: b) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane	76

		według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych.	
6.3.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) 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: b) 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.	30
6.4.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) 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: b) 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.	24
7.1.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, 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: c) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	69
7.2.	II. Programowanie i rozwiązywanie problemów	Zdający: II.4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami	69

	z wykorzystaniem komputera i innych urządzeń cyfrowych.	w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, 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: c) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	
7.3.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, 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: c) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	28
7.4.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy, tworzy i modyfikuje formularze oraz raporty, 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: c) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy.	27
7.5.	II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.	Zdający: II.4) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel oraz sieciową aplikację bazodanową dla danych związanych z rozwiązywanym problemem, formułuje kwerendy […], stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji […].	33

WYKRES 2. POZIOM WYKONANIA ZADAŃ W OBSZARZE WYMAGAŃ OGÓLNYCH



Komentarz (dotyczy wyników krajowych)

W roku 2023 po raz pierwszy odbył się egzamin maturalny z informatyki w formule 2023. W stosunku do arkusza egzaminacyjnego z lat poprzednich zmianie uległa nie tylko szata graficzna, ale przede wszystkim forma tego egzaminu. Po raz pierwszy arkusz nie był podzielony na dwie części, w związku z czym zdający mieli dostęp do autonomicznego stanowiska komputerowego podczas całego egzaminu.
Do egzaminu przystąpiło 5103 absolwentów liceów ogólnokształcących. Uzyskali oni średni wynik 45%.

Analiza jakościowa zadań

Tegoroczny arkusz składał się z 23 zadań. 21 zadań ujęto w 5 wiązek tematycznych, pozostałe dwa zadania były zadaniami samodzielnymi. Dwa zadania okazały się dla zdających bardzo trudne (poziom wykonania[5] mniejszy niż 19%), 10 zadań było dla zdających trudnych (poziom wykonania każdego z tych zadań mieścił się w zakresie od 20% do 49%), 6 zadań okazało się umiarkowanie trudne (poziom wykonania w zakresie od 50% do 69%), a 5 zadań było łatwych (poziom wykonania od 70% do 89%). Zadań o poziomie wykonania powyżej 89% (czyli bardzo łatwych) nie było w arkuszu.

Rozkład punktacji na poszczególnych poziomach trudności wyglądał następująco: za zadania bardzo trudne można było uzyskać w sumie 5 punktów (10% maksymalnej liczby punktów do zdobycia), za zadania trudne można było uzyskać w sumie 25 punktów (50% maksymalnej liczby punktów do zdobycia), za zadania umiarkowanie trudne można było uzyskać w sumie 12 punktów (24% maksymalnej liczby punktów do zdobycia), a za zadania łatwe można było uzyskać w sumie 8 punktów (16% maksymalnej liczby punktów do zdobycia).

Zadania praktyczne stanowiły zdecydowaną większość zadań, a za ich rozwiązanie można było zdobyć w sumie 35 punktów, czyli aż 70% wszystkich możliwych punktów do zdobycia. Za poprawne rozwiązanie każdego z tych zadań zdający otrzymywali najwyższą ocenę tylko wówczas, gdy zapisana poprawna odpowiedź w pliku tekstowym wynikała również z komputerowej realizacji zadania. Brak plików zawierających komputerową realizację zadań był jednoznaczny z negatywną oceną rozwiązań.

Zadania, z którymi zdający poradzili sobie najsłabiej

Najtrudniejsze dla zdających okazały się: zadanie 3.3. oraz zadanie 3.4. (poziom wykonania tych zadań to odpowiednio 7% i 8%). Poprawne rozwiązanie tych zadań wymagało od zdających napisania programu analizującego 10 000 kolejnych cyfr rozwinięcia dziesiętnego liczby π i wyszukania wśród nich ciągów spełniających podane kryteria. Zadania te zostaną szczegółowo omówione w dalszej części komentarza.

W grupie zadań trudnych znalazły się zadania: 6.4. (poziom wykonania – 24%), 6.3. (poziom wykonania – 28%), 2.5. (poziom wykonania – 29%), 7.3. (poziom wykonania – 34%), 2.1. (poziom wykonania – 35%), 7.5. (poziom wykonania – 37%), 7.4. (poziom wykonania –

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

39%), 3.2. (poziom wykonania – 40%), 2.3. i 2.4. (poziom wykonania – po 46%). Zadanie te sprawdzały różne umiejętności. Są wśród nich zadania programistyczne, zadania wymagające wykonania symulacji z użyciem arkusza kalkulacyjnego, zadania bazodanowe i zadanie wymagające zapisania algorytmu w postaci pseudokodu lub w języku programowania. Większość z nich (poza zadaniami 2.1., 7.5. i 2.4.) to zadania praktyczne.

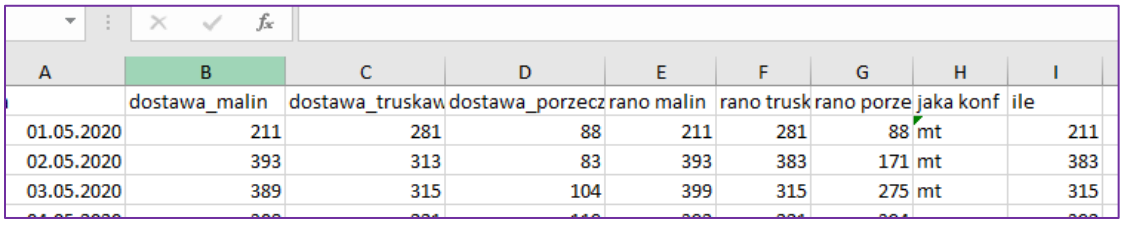
Zawarta w arkuszu egzaminacyjnym wiązka zadań zawierająca zadania od 6.1. do 6.4., to zadania przeznaczone przede wszystkim do wykonania w arkuszu kalkulacyjnym (napisanie programu lub programów dających odpowiedzi do tych zadań jest również możliwe, jednak rozwiązanie tego typu zadań w arkuszu kalkulacyjnym jest szybsze i łatwiejsze). Spośród tych zadań dwa pierwsze sprawdzały umiejętność posługiwania się podstawowymi narzędziami i funkcjami dostępnymi w arkuszu kalkulacyjnym. Dwa kolejne zadania, które okazały się trudne dla zdających (poziom wykonania odpowiednio 24% i 28%), wymagały umiejętności przeprowadzenia symulacji z pomocą arkusza kalkulacyjnego. Cała wiązka oparta była na pliku z danymi dotyczącymi dostaw owoców do fikcyjnej przetwórni produkującej konfitury.

Zadania 6.3. i 6.4. wymagały wykonania symulacji produkcji konfitur w fikcyjnej przetwórni. Zgodnie z treścią zadania do przetwórni codziennie dostarczano owoce trzech rodzajów. Przetwórnia produkowała trzy rodzaje konfitur złożonych jedynie z dwóch rodzajów owoców. Decyzję o tym, które konfitury będą produkowane podejmowano na podstawie ilości owoców dostępnych w przetwórni danego dnia po dostawie. Do produkcji używano dwóch rodzajów owoców, których było najwięcej danego dnia w przetwórni. Niewykorzystane owoce pozostawały do wykorzystania w następnym dniu. W zadaniu 6.3. należało obliczyć i podać liczbę dni, w których produkowane były konfitury każdego rodzaju, a w zadaniu 6.4. – ile kilogramów każdego rodzaju konfitur wyprodukowano.

Najlepszym sposobem wykonania symulacji jest utworzenie dodatkowych kolumn danych, w których zapisywany jest stan magazynowy każdego rodzaju owoców każdego dnia oraz informacja o rodzaju konfitury, jaka danego dnia będzie produkowana.

Przykładowe rozwiązanie zadań 6.3. i 6.4.przedstawiono poniżej:

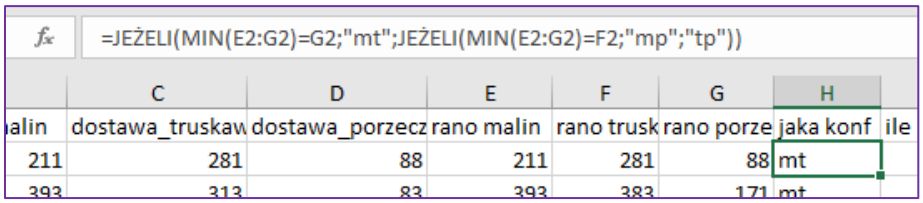
1. Należy utworzyć kolumny, w których zapisany zostanie stan poszczególnych owoców rano po dostawie („rano maliny”, „rano truskawki”, „rano porzeczki”) oraz kolumnę z informacją o rodzaju konfitur „jaka konfitura” i ilości użytych owoców „ile”) (rysunek 1.).



	A	B	C	D	E	F	G	H	I
		dostawa_malin	dostawa_truskaw	dostawa_porzecz	rano malin	rano trusk	rano porze	jaka konf	ile
	01.05.2020	211	281	88	211	281	88	mt	211
	02.05.2020	393	313	83	393	383	171	mt	383
	03.05.2020	389	315	104	399	315	275	mt	315

Rysunek 1. Przykładowe rozwiązanie zadań 6.3. i 6.4. (krok 1.).

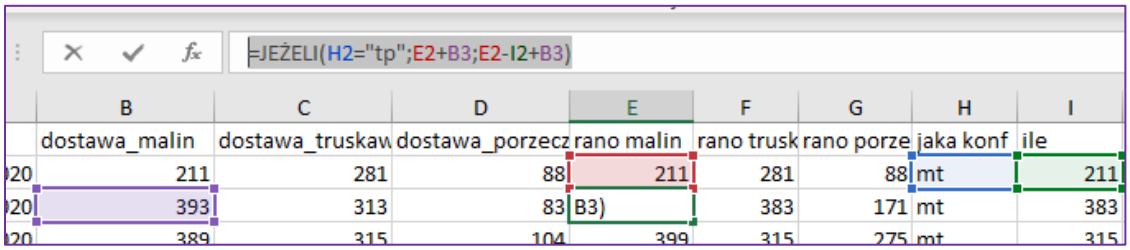
2. W kolumnie oznaczającej, która konfitura będzie produkowana (na rysunku 2. kolumna H), należy wpisać formułę, która określa rodzaj konfitur na podstawie ilości owoców rano np.: =JEŻELI(MIN(E2:G2)=G2;"mt";JEŻELI(MIN(E2:G2)=F2;"mp";"tp"))
(zapis oznacza, że jeżeli najmniej jest porzeczek, to produkowana jest konfitura malinowo-truskawkowa itd.)



Rysunek 2. Przykładowe rozwiązanie zadań 6.3. i 6.4. (krok 2.).

3. W kolumnie oznaczającej ilość konfitur należy wpisać formułę, która wyliczy, jaka ilość owoców zostanie użyta. Należy wziąć pod uwagę owoce tylko jednego rodzaju, ponieważ w zadaniu jest informacja, że owoce do produkcji brane są w stosunku 1:1 oraz że na produkcję jednego kilograma konfitur używa się 2 kg owoców (czyli konfitur powstaje tyle samo, ile używa się do ich produkcji owoców jednego rodzaju) – formuła wyliczająca ilość może wyglądać następująco:
=JEŻELI(H2="mt";MIN(E2:F2);JEŻELI(H2="mp";MIN(E2;G2);MIN(F2:G2)))
(zapis oznacza, że jeżeli produkujemy konfitury malinowo truskawkowe, należy wziąć pod uwagę minimum z malin i truskawek itd.).

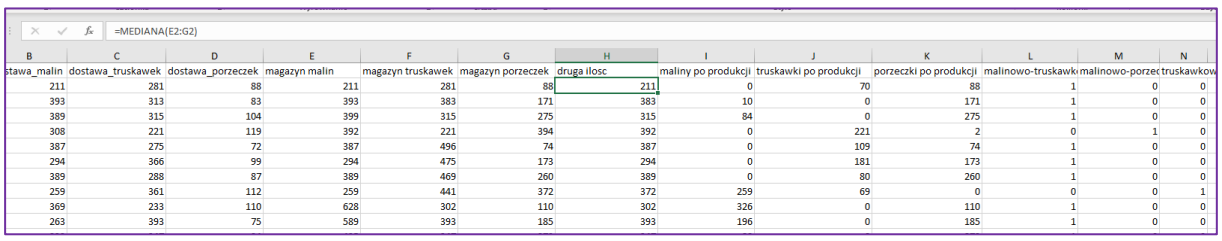
4. W kolumnie E, F i G w wierszu trzecim należy wpisać formuły wyliczające stan owoców rano po dostawie i produkcji poprzedniego dnia np. dla malin:
=JEŻELI(H2="tp";E2+B3;E2-I2+B3)
(zapis oznacza, że jeżeli nie używano malin, czyli produkowane są konfitury truskawkowo-porzeczkowe, to należy wpisać poprzedni stan rano plus dostawę, w przeciwnym wypadku należy wpisać poprzedni stan minus ilość użytych owoców plus dostawę).



Rysunek 3. Przykładowe rozwiązanie zadań 6.3. i 6.4. (krok 4.).

5. Po skopiowaniu wszystkich formuł w dół zostanie otrzymana pełna symulacja. W celu uzyskania prawidłowej odpowiedzi do zadania 6.3. wystarczy zliczyć z kolumny H, ile było produkowanych konfitur każdego rodzaju. Zsumowanie wartości z kolumny I w zależności od wartości w kolumnie H (np. z pomocą formuły SUMA.JEŻELI) pozwala uzyskać prawidłową odpowiedź do zadania 6.4.

Na rysunku 4. przedstawiono fragment poprawnego rozwiązania zdającego, który w kolejnych kolumnach obliczał stan trzech rodzajów owoców w magazynie, oddzielnie przed i po użyciu ich do produkcji. W celu określenia rodzaju produkowanej w danym dniu konfitury obliczał wartość środkową (medianę) trzech ilości owoców, które rano znajdowały się w magazynie.



Rysunek 4. Fragment poprawnego rozwiązania.

Cztery z pięciu zadań, ujętych w wiązkę dotyczącą **liczb zapisanych w systemie binarnym**, okazały się być zadaniami trudnymi dla zdających. Wyjątek stanowiło zadanie 2.2., które zostało wykonane na poziomie 52%. Rozwiązania zadań 2.1. i 2.4. należało zapisać w arkuszu, natomiast pozostałe zadania z tej wiązki to zadania praktyczne.

Rozwiązanie zadania 2.1. (poziom wykonania – 35%) wymagało zapisania algorytmu obliczającego, z ilu bloków składa się zapis binarny podanej liczby całkowitej.

Definicję bloku wraz z przykładami umieszczono na początku zadania:

Zadanie 2. Liczby binarne

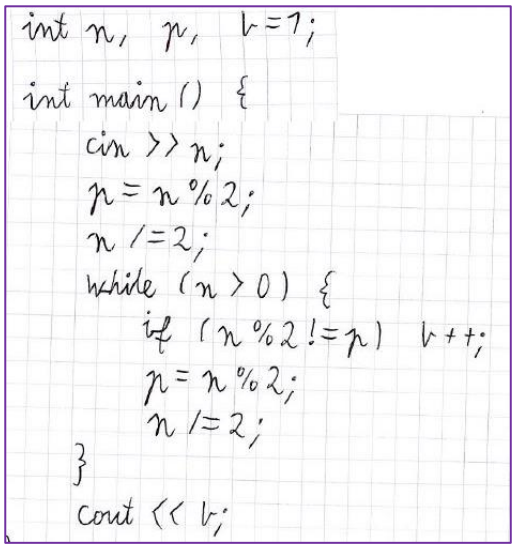
W tym zadaniu rozważamy binarny zapis liczb całkowitych dodatnich.
Blokiem w zapisie binarnym liczby nazywamy każdy niepusty, maksymalny (nie można go rozszerzyć ani z lewej, ani z prawej strony) ciąg kolejnych takich samych cyfr w tym zapisie.

Przykład:
Liczba binarna 111110000110111 składa się z pięciu *bloków* – trzech *bloków* złożonych z jedynek (11111, 11 i 111) i dwóch *bloków* złożonych z zer (0000 i 0).
Liczba binarna 1111111111111111 składa się z jednego *bloku* złożonego z jedynek.

Zgodnie z zapisaną w zadaniu 2.1. specyfikacją dana była liczba całkowita dodatnia, a oczekiwany wynik, to liczba bloków w zapisie binarnym danej liczby. Aby policzyć liczbę bloków występujących w zapisie binarnym takiej liczby, należało najpierw otrzymać kolejne cyfry zapisu binarnego tej liczby. Analizując zadanie, warto zwrócić uwagę, że zapis podanej liczby w postaci binarnej nie jest konieczny do prawidłowego zapisu algorytmu. Zauważenie, że liczba bloków jest równa powiększonej o jeden liczbie zmian kolejnych reszt z dzielenia liczby przez dwa (z 1 na 0 lub odwrotnie), które tworzą zapis binarny liczby, umożliwiało otrzymanie poprawnego rozwiązania zadania z pominięciem zamiany zapisanej w systemie dziesiętnym danej liczby na zapis w systemie dwójkowym.

Wielu zdających jednak najpierw zapisywało postać binarną liczby, by w następnym kroku ją analizować – utrudniało to rozwiązanie zadania i powodowało powstawanie dodatkowych okazji do popełniania błędów.

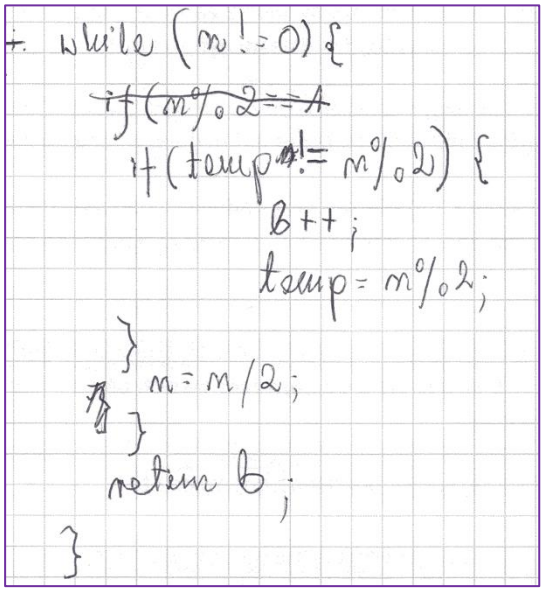
Na rysunku 5. przedstawiono przykładowe poprawne rozwiązanie zdającego.



```
int n, p, b=1;
int main() {
    cin >> n;
    p = n % 2;
    n /= 2;
    while (n > 0) {
        if (n % 2 != p) b++;
        p = n % 2;
        n /= 2;
    }
    cout << b;
```

Rysunek 5. Przykładowe rozwiązanie zadania 2.1.

W algorytmie przedstawionym na rysunku 6. pominięto inicjację zmiennych. W zależności od przyjętego pomysłu na realizację kolejnych kroków algorytmu do zmiennej *temp* powinna zostać początkowo podstawiona reszta z dzielenia podanej liczby *n* przez dwa (*n*%2) lub dowolna wartość różna od 1 i 0. Brakuje także inicjacji zmiennej *b,* która z uwagi na definicję bloku powinna przyjąć wartość początkową równą 1. W tego typu zadaniach wymagających zapisania algorytmu, nie należy pomijać inicjacji zmiennych, ponieważ ma ona znaczący wpływ na poprawność wyniku zwracanego przez algorytm. W tym zadaniu jest to tym bardziej istotne, ponieważ bardzo często w przypadku użycia kompilatora, w którym pewne zmienne typu liczbowego są inicjowane, to zazwyczaj przypisywana jest im wartość 0.



Rysunek 6. Niepoprawne rozwiązanie zadania 2.1.

W zadaniu 2.3. (poziom wykonania – 46%) należało napisać program komputerowy, który spośród stu liczb, zapisanych w pliku bin.txt, odnajdzie i wypisze największą z nich. Zgodnie z warunkami zadania liczby w tym pliku zapisane zostały w postaci binarnej, każda z nich była większa od 0 oraz zapis każdej z liczb zaczynał się od cyfry 1. Aby wyszukać największą liczbę (także w postaci binarnej) można było zamienić każdą z liczb w pliku na liczbę w zapisie dziesiętnym, znaleźć największą z nich, a następnie zamienić ponownie jej zapis dziesiętny na binarny. Nie wszyscy zdający zauważyli, że można było także potraktować zapisy binarne liczb jako teksty – wtedy, porównując zapis dwóch liczb różnych długości, większą z nich jest ta, której zapis jest dłuższy. W przypadku gdy obydwa zapisy są tej samej długości można porównać liczby jako teksty – zapis np. „11000” jest w kolejności alfabetycznej większy niż „10100”.

W zadaniu 2.4. (poziom wykonania – 46%) należało wykonać obliczenia z podaną i opisaną funkcją XOR.

Zadanie 2.4. (0–1)

Dla nieujemnych liczb całkowitych *a* i *b* wynikiem operacji *a* XOR *b* jest liczba, której kolejne bity są wyliczane na podstawie poniższej tabelki z odpowiadających sobie bitów w zapisie binarnym liczb *a* i *b*. Jeśli jeden zapis jest krótszy od drugiego, to uzupełniamy go zerami z lewej strony (na najbardziej znaczących pozycjach).

p	*q*	*p* XOR *q*
1	1	0
1	0	1
0	1	1
0	0	0

np.
4_{10} XOR 7_{10} = 100_2 XOR 111_2 = 011_2 = 3_{10}
6_{10} XOR 11_{10} = 0110_2 XOR 1011_2 = 1101_2=13_{10}

Oblicz (123_{10} XOR 101101_2) XOR $2D_{16}$. Wynik podaj w systemie **dziesiętnym**.

Rysunek 7. zadanie 2.4.

Zgodnie z opisem zapis wszystkich liczby należało zamienić na zapis binarny. Zdający prawidłowo wykonujący obliczenia otrzymywali następuje działania:

(1111011_2 XOR 0101101_2) XOR 101101_2 = 1010110_2 XOR 0101101_2 = 1111011_2 = 123_{10}

Najtrudniejsze w tej wiązce okazało się zadanie 2.5. (poziom wykonania – 29%), w którym należało dla każdej liczby binarnej *p* z pliku bin.txt wykonać operację *p* XOR (*p* div 2) (gdzie *p* div 2 – oznacza dzielenie całkowite *p* przez 2) i otrzymane wyniki zapisać w postaci binarnej do pliku.

Można było wykonać to zadanie, najpierw zamieniając każdą z liczb binarnych zapisanych w pliku na liczbę w zapisie dziesiętnym, następnie obliczyć połowę tej liczby zaokrągloną w dół do liczby całkowitej (*p* div 2) i ponownie zamienić zapis liczby na zapis binarny, aby ostatecznie wykonać operację XOR na kolejnych bitach dwóch liczb: liczby oryginalnej oraz liczby otrzymanej w wyniku opisanych przekształceń. W ten sposób postępowała większość zdających. Na rysunkach 8. i 9. pokazano fragment rozwiązania zdającego.

```
73    string funkcjax(string s1, string s2)
74    {
75        int d;
76        int g;
77        string nowy="";
78        if(s1.size()>s2.size())
79        {
80            d=s2.size();
81            g=s1.size();
82            for(int i=d;i<s1.size();i++)
83            {
84                s2='0'+s2;
85            }
86        }
87        else
88        {
89            d=s1.size();
90            g=s2.size();
91            for(int i=d;i<s1.size();i++)
92            {
93                s1='0'+s1;
94            }
95        }
96        for(int i=0;i<g;i++)
97        {
98            if(s1[i]==s2[i]) nowy+='0';
99            else nowy+='1';
100       }
101       return nowy;
102   }
103   int na_dziesietny(string s)
104   {
105       int x=0;
106       int sys=1;
107       for(int i=s.size()-1;i>=0;i--)
108       {
109           x+=(s[i]-'0')*sys;
110           sys*=2;
111       }
112       return x;
```

Rysunek 8. Przykładowe rozwiązanie 2.5.

```
fstream zapis;
zapis.open(nazwa,ios::out);
for(int i=0;i<k;i++)
{
    zapis<<funkcjax(bin[i],z_dzies_na_binarny(na_dziesietny(bin[i])/2))<<endl;
}
zapis.close();
```

Rysunek 9. Przykładowe rozwiązanie 2.5. – dalszy ciąg.

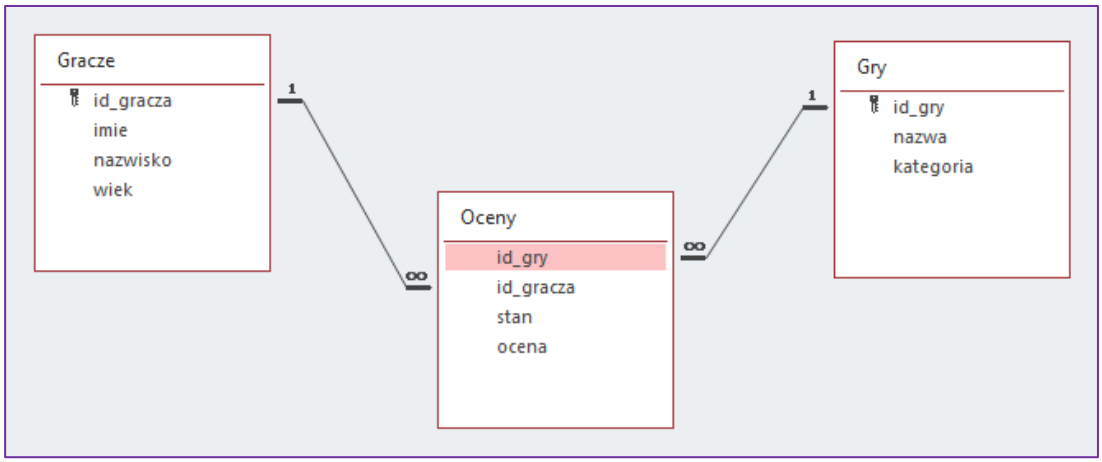
Zadanie także można było rozwiązać, nie przeliczając zapisów binarnych na dziesiętne. Można zauważyć, że jeśli mamy daną liczbę w zapisie binarnym, to podzielenie jej przez 2 oznacza przesunięcie wszystkich bitów w prawo o 1 miejsce, np.:

liczba 123 w zapisie binarnym to 1111011, a po podzieleniu całkowitym przez 2 jest to 61, czyli 111101 (jak widać wystarczyło skasować ostatnią jedynkę).

Ponieważ liczba podzielona przez 2 w zapisie binarnym ma jeden znak mniej, przed wykonaniem operacji XOR należy na początku jej zapisu dopisać cyfrę 0, czyli w przykładzie wyżej otrzymamy zapis 0111101. Teraz wykonujemy operację XOR.

Trudne dla zdających okazały się też trzy zadania: 7.3. (poziom wykonania – 34%), 7.5. (poziom wykonania – 37%) i 7.4. (poziom wykonania – 39%), z wiązki zadań wymagającej utworzenia bazy danych i wykonania zapytań, które dadzą odpowiedzi na poszczególne pytania. Dwa z nich to zadania praktyczne, natomiast zadanie 7.5. polegało na zapisaniu w arkuszu egzaminacyjnym zapytania w języku SQL.

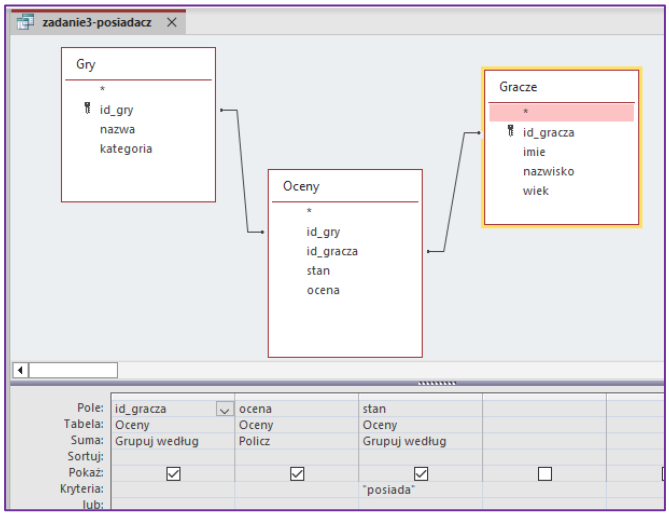
Wszystkie zadania z tej wiązki opierają się na danych dotyczących gier planszowych. Dane zapisane są w trzech plikach tekstowych, które należy zaimportować do trzech tabel w wybranym narzędziu bazodanowym (może to być MS Access, Base z Open/Libre Office lub MySQL). Baza, po poprawnym zaimportowaniu danych, powinna się składać z trzech tabel: Gry, Gracze i Oceny, w których przechowywane są informacje o danych graczy, grach planszowych i ocenach tych gier wystawianych przez graczy (rysunek 10.).



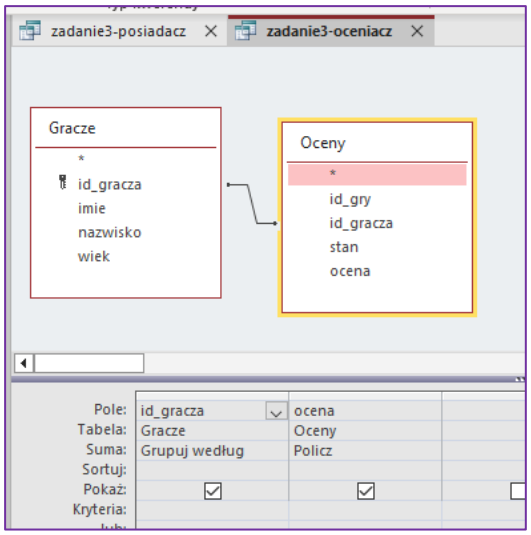
Rysunek 10. Baza danych w zadaniu 7.

W zadaniu 7.3. należało podać liczbę graczy, którzy wystawili co najmniej jedną ocenę, ale nie posiadają (nie mają stanu „posiada”) żadnej z ocenianych gier. Aby rozwiązać to zadanie, trzeba od liczby wszystkich graczy, którzy wystawili jakąkolwiek ocenę, odjąć liczbę tych, którzy posiadają którąkolwiek z ocenianych gier. Warto zauważyć, że w bazie nie ma informacji o stanie posiadania innych gier niż te, które zostały przez graczy ocenione.

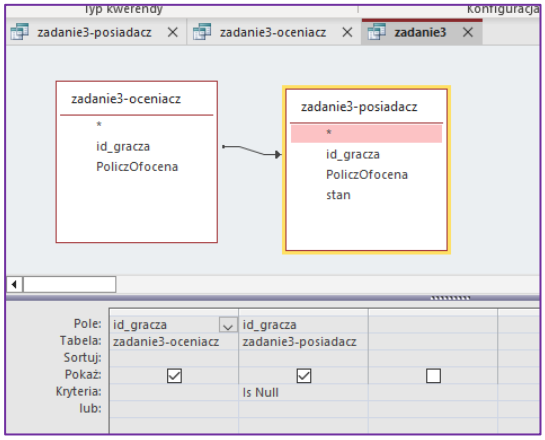
Przykładowe rozwiązanie zadania 7.3. przedstawiono na rysunkach 11., 12. i 13.



Rysunek 11. Pierwsza kwerenda pomocnicza do zadania 7.3.



Rysunek 12. Druga kwerenda pomocnicza do zadania 7.3.

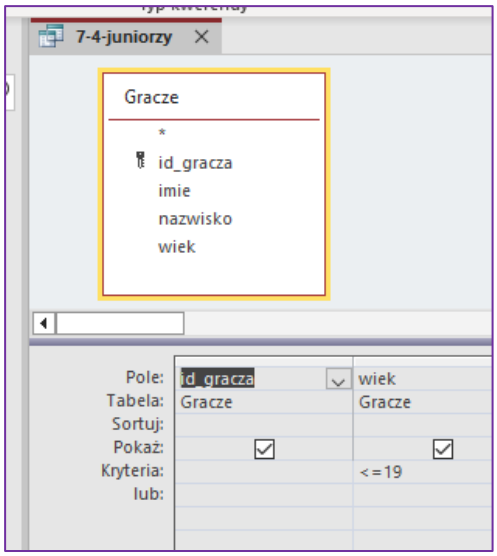


Rysunek 13. Przykładowe rozwiązanie zadania 7.3. wykonane przez zdającego z wykorzystaniem dwóch kwerend pomocniczych.

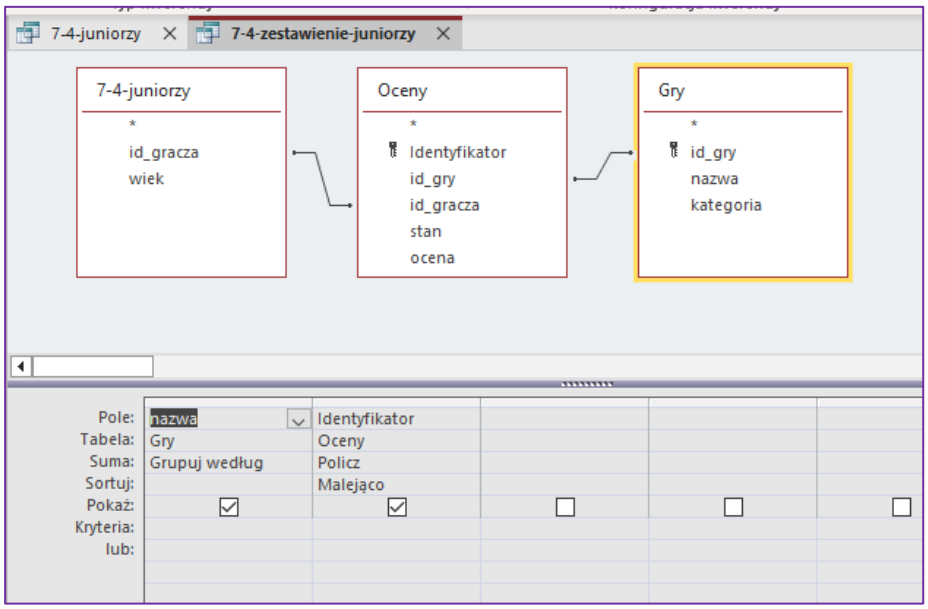
W zadaniu 7.4. osoby oceniające gry podzielono na trzy kategorie wiekowe: juniorzy (do 19 lat), seniorzy (od 20 do 49 lat) oraz weterani (od 50 lat). Należało wykonać zestawienie, w którym dla każdej kategorii wiekowej zostanie podana największa liczba ocen wystawionych jednej grze przez użytkowników z tej kategorii wiekowej oraz nazwy gier z tą liczbą ocen. Jeżeli gier, które otrzymały taką samą największą liczbę ocen od użytkowników z danej kategorii wiekowej, jest więcej niż jedna, należało podać tytuły ich wszystkich.

Wykonując to zadanie, można było utworzyć pole oznaczające kategorię wiekową i następnie się nią posługiwać, jednak większość zdających oddzielnie zliczała liczby ocen gier dla każdej kategorii wiekowej, tworząc minimum trzy oddzielne kwerendy.

Rysunki 14. i 15. przedstawiają fragmenty rozwiązania zadania 7.4.

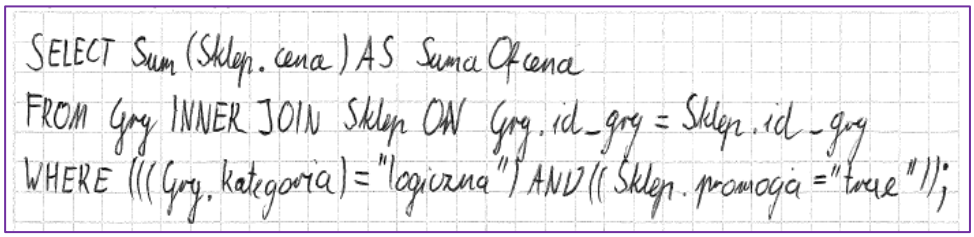


Rysunek 14. fragment przykładowego rozwiązania



Rysunek 15. fragment przykładowego rozwiązania

W zadaniu 7.5. opisano sytuację, w której do podanych tabel dołączono kolejną reprezentującą cennik sklepu z grami komputerowymi. W tym zadaniu należało w arkuszu egzaminacyjnym zapisać zapytanie w języku SQL, w wyniku którego zostanie otrzymany łączny koszt zakupu w tym sklepie po jednej sztuce ze wszystkich gier z kategorii „logiczna” dostępnych w cenie promocyjnej. Przykładowe poprawne rozwiązanie pokazano na rysunku 16.



Rysunek 16. Przykładowe rozwiązanie zadania 7.5.

Zadania, z którymi zdający poradzili sobie najlepiej

Zadaniami łatwymi dla zdających okazały się zadania: 5. (poziom wykonania – 78%), 6.1. i 6.2. (poziom wykonania odpowiednio – 81% i 79%) oraz 7.1. i 7.2. (poziom wykonania odpowiednio – 77% i 74%). Zadanie 5. wymagało znajomości zapisu pozycyjnego liczb, zadanie 6.1. i 6.2. były zadaniami sprawdzającymi podstawową znajomość narzędzi i funkcji arkusza kalkulacyjnego, natomiast 7.1. i 7.2. wymagały umiejętności tworzenia prostych kwerend w bazie danych.

Zdający dość dobrze poradzili sobie także z zadaniami 1.1., 1.2. i 1.3. (poziom wykonania odpowiednio – 62%, 61% i 63%), które wymagały rozumienia pojęcia rekurencji. Umiarkowanie trudne było także zadanie 4. (poziom wykonania – 54%) z zakresu bezpieczeństwa danych i szyfrowania informacji oraz zadania wymagające napisania prostego programu 3.1. (poziom wykonania – 55%) i 2.2. (poziom wykonania – 52%).

Problem pod lupą – „Liczba Pi" – zadania programistyczne 3.2., 3.3. i 3.4.

Zadania 3.2., 3.3. i 3.4. (poziom wykonania odpowiednio – 40%, 7% i 8%) należą do wiązki zadań programistycznych, w których operowano na danych zapisanych w pliku pi.txt. Plik ten zawierał 10 000 cyfr rozwinięcia dziesiętnego liczby π zapisanych w oddzielnych wierszach. Do całej wiązki zadań 3.1.–3.4. dołączono także plik przykładowy zawierający pierwsze 100 wierszy z pliku pi.txt. Pod każdym z zadań umieszczono odpowiedzi dotyczące tego pliku, aby zdający mogli przetestować poprawność działania swoich programów.

W zadaniu 3.2. należało znaleźć fragmenty dwucyfrowe złożone z par kolejnych cyfr z pliku pi.txt, które występują w pliku najczęściej oraz te, które występują najrzadziej. Zadanie to okazało się zadaniem dość trudnym dla zdających, mimo, że jest zadaniem klasycznym. Jednym z prostszych pomysłów na rozwiązanie tego zadania jest utworzenie 100-elementowej tablicy, w której każde pole będzie przechowywać liczbę fragmentów dwucyfrowych o wartości liczbowej równej indeksowi danego pola tablicy. Początkowo wszystkie pola tablicy należy wypełnić zerami. W kolejnym kroku odczytywane są kolejne dwie cyfry z pliku, zamieniane są one na liczbę dwucyfrową (pierwsza z nich jest cyfrą dziesiątek, druga – cyfrą jedności), a następnie wartość w polu tablicy o indeksie równym otrzymanej liczbie zostaje zwiększona o 1. Po wypełnieniu tablicy liczbą wystąpień poszczególnych fragmentów, należy odszukać w niej wartość największą i wartość najmniejszą.

Przykładowe rozwiązanie zdającego pokazano na rysunku 17.

```
int t[101];
wyzeruj(t,101);
for(int i=0;i<k-1;i++)
{
    t[pi[i]*10+pi[i+1]]=t[pi[i]*10+pi[i+1]]+1;
}
int najmniejsza=t[0];
int najmn=0;
int najw=0;
string najww;
string najmm;
int najwieksza=t[0];
for(int j=1;j<100;j++)
{
    if(t[j]>najwieksza)
    {
        najwieksza=t[j];
        najw=j;
    }
    if(t[j]<najmniejsza)
    {
        najmniejsza=t[j];
        najmn=j;
    }
}
najww=to_string(najw);
if(najww.size()==1) najww='0'+najww;
najmm=to_string(najmn);
if(najmm.size()==1) najmm='0'+najmm;
zapis<<najww<<" "<<najwieksza<<endl<<najmm<<" "<<najmniejsza<<endl;
zapis.close();
```

Rysunek 17. Zadanie 3.2.

Do najtrudniejszych w arkuszu zadań 3.3. i 3.4., zamieszczono dodatkową informację zawierającą definicję pojęcia ciągu rosnąco-malejącego, którego te zadania dotyczyły.

Informacja do zadań 3.3. i 3.4.

Skończony co najmniej 4-elementowy ciąg liczb ($a_1, a_2, \ldots, a_n$) jest *rosnąco-malejący*, jeśli można podzielić go na dwa ciągi, z których pierwszy jest rosnący, a drugi – malejący, tzn. jeśli istnieje takie $k \in \{2, 3, \ldots, n-2\}$, że $a_1 < a_2 < \ldots < a_k$ oraz $a_{k+1} > a_{k+2} > \ldots > a_n$.

Przykład:
Ciąg (2, 5, 7, 9, 8, 3, 1) jest *rosnąco-malejący*, bo można go podzielić na dwa ciągi: rosnący (2, 5, 7) i malejący (9, 8, 3, 1) lub – odpowiednio – (2, 5, 7, 9) i (8, 3, 1). Ciąg (5, 9, 9, 4,1) także jest *rosnąco-malejący*.
Przykłady ciągów, które nie są *rosnąco-malejące*, to: (2, 5, 8, 4, 3, 4, 5), (1, 2, 3, 4), (5, 5, 3, 2, 1).

Rysunek 18. Informacja do zadań 3.3. i 3.4.

Jak wynika z definicji ciąg rosnąco-malejący, to ciąg, który można podzielić na dwa ciągi, z których pierwszy jest rosnący, a drugi jest malejący. Przy tym wprost z definicji wynika, że każdy z tych ciągów musi mieć przynajmniej dwa elementy (ponieważ liczba k należy do przedziału od 2 do n-2, gdzie n oznacza długość całego ciągu). Warto też zauważyć, że w definicji nie określono, jaka jest zależność między elementami a_k oraz a_{k+1}. Dla wyjaśnienia zostały umieszczone przykłady wskazujące m.in., że a_k może być np. równe a_{k+1}, a także, dodatkowo, przypominające, że ciąg malejący jak i rosnący muszą mieć przynajmniej 2 elementy.

W zadaniu 3.3. należało policzyć, ile w pliku pi.txt występuje ciągów rosnąco-malejących, złożonych z kolejnych cyfr zapisanych w tym pliku, o długości równej 6. Rozwiązując to zadanie można było wybrać dwie główne strategie:

- 1 sposób rozwiązania to ustalenie wszystkich możliwych przypadków, w których 6-elementowy ciąg jest rosnąco-malejący i sprawdzanie, czy każde kolejne sześć cyfr z pliku należy do któregoś z tych przypadków.
  Zwróćmy uwagę, że istnieją 3 podstawowe możliwości podziału ciągu 6-elementowego na dwa ciągi, w którym pierwszy jest rosnący i drugi jest malejący:
  a) 2-elementowy ciąg rosnący i 4-elementowy ciąg malejący (czyli $a_1<a_2$ oraz $a_3>a_4>a_5>a_6$)
  b) 3-elementowy ciąg rosnący i 3-elementowy ciąg malejący (czyli $a_1<a_2<a_3$ oraz $a_4>a_5>a_6$)
  c) 4-elementowy ciąg rosnący i 2-elementowy ciąg malejący (czyli $a_1<a_2<a_3<a_4$ oraz $a_5>a_6$)
  Zauważmy, że z przypadku b), w zależności od tego, czy a_3 jest większe od a_4, czy też mniejsze, otrzymujemy przypadek a) albo c), dlatego należy w przypadku b) dopisać $a_3 = a_4$ (tylko wówczas ciągu nie da się podzielić na część rosnącą i malejącą inaczej niż w proporcji 3 elementy na 3 elementy). Zależność między a_2 i a_3 w przypadku a) oraz a_4 i a_5 w przypadku c) nie ma znaczenia. Czyli ostatecznie otrzymujemy możliwości:
  a) 2-elementowy ciąg rosnący i 4-elementowy ciąg malejący (czyli $a_1<a_2$ oraz $a_3>a_4>a_5>a_6$)
  b) 3-elementowy ciąg rosnący i 3-elementowy ciąg malejący (czyli $a_1<a_2<a_3=a_4>a_5>a_6$)
  c) 4-elementowy ciąg rosnący i 2-elementowy ciąg malejący (czyli $a_1<a_2<a_3<a_4$ oraz $a_5>a_6$)
  Można także rozpisać przypadki a) i c), dodatkowo uwzględniając zależności między elementami a_2 i a_3 w przypadku a) oraz między elementami a_4 i a_5 w przypadku c) – otrzymamy wtedy łącznie 7 różnych możliwości utworzenia 6-elementowego ciągu rosnąco-malejącego

- 2 sposób rozwiązania to szukanie ciągów rosnąco malejących ze sprawdzaniem, czy są 6-elementowe, to jest dla każdej cyfry z pliku (do cyfry w wierszu 9995) sprawdzamy, czy jest ona pierwszym elementem ciągu rosnącego i następnie sprawdzamy kolejne cyfry aż trafimy na koniec ciągu rosnącego, w następnym kroku sprawdzamy, czy następne cyfry tworzą ciąg malejący. Jeśli otrzymany w ten sposób ciąg rosnąco-malejący jest 6-elementowy, to zliczamy go do wyniku. Jest to podejście trudniejsze z przynajmniej dwóch powodów:
  a) musimy pamiętać, że szukamy ciągu o konkretnej długości, a więc np. zliczać, który element sprawdzamy
  b) jeśli trafimy na ostatnią cyfrę w ciągu rosnącym, to ciąg malejący może się zaczynać od cyfry następnej po niej (jeśli następna cyfra jest równa aktualnej) albo jednocześnie nasza aktualna cyfra może być początkiem ciągu malejącego – o tym przypadku często zapominali zdający.

Na rysunku 19. przedstawiono fragment niepoprawnego rozwiązania, w którym sprawdzane są zależności pomiędzy elementami każdego 7-elementowego ciągu z pominięciem elementu środkowego.

```
ile_rosn = 0
for gp in range(0, 9994):
    if pi[gp] < pi[gp+1] < pi[gp+2] and pi[gp+4] > pi[gp+5] > pi[gp + 6]:
        ile_rosn +=1
print(ile_rosn)
```

Rysunek 19. Fragment rozwiązania, w którym nie uwzględniono kilku warunków.

Rysunek 20. przedstawia fragment rozwiązania innego zdającego (strategia 2), który nie sprawdził czy ciąg, który zlicza zaczyna się ciągiem rosnącym (zdający od *i*-tego indeksu przesuwa się wzdłuż ciągu rosnącego, a potem do końca ciągu malejącego – nie sprawdza, czy ciąg rosnący ma przynajmniej dwa elementy). W wyniku tego postępowania zdający zliczył np. ciągi minimum 6-elementowe, które w całości są malejące.

```
60      int k;
61      int l;
62      int m;
63      int result3_3=0;
64      for(int i=0;i<p;i++)
65      {
66          k=l;
67          l=i+1;
68          while(l<p&&(k<4&&input[l]>input[l-1]))
69          {
70              l++;
71              k++;
72          }
73          m=l;
74          l++;
75          while(l<p&&input[l]<input[l-1])
76          {
77              l++;
78              m++;
79          }
80          if(m+k>=6)
81              result3_3++;
82
83      }
```

Rysunek 20. Fragment rozwiązania zadania 3.3. z błędem.

Niepoprawny sposób realizacji pierwszej strategii rozwiązania zadania pokazano na rysunku 21. Wśród sprawdzanych warunków nie uwzględniono przypadków, w których element znajdujący się na końcu ciągu rosnącego jest równy pierwszemu elementowi ciągu malejącego.

```
#3
odp3 = 0
for i in range(len(dane)-6):
    if int(dane[i]) > int(dane[i+1]) and int(dane[i+1]) > int(dane[i+2]) and int(dane[i+2]) > int(dane[i+3]) and int(dane[i+3]) > int(dane[i+4]) and int(dane[i+4]) < int(dane[i+5]):
        odp3 += 1
    if int(dane[i]) > int(dane[i+1]) and int(dane[i+1]) > int(dane[i+2]) and int(dane[i+2]) > int(dane[i+3]) and int(dane[i+3]) < int(dane[i+4]) and int(dane[i+4]) < int(dane[i+5]):
        odp3 += 1
    if int(dane[i]) > int(dane[i+1]) and int(dane[i+1]) > int(dane[i+2]) and int(dane[i+2]) < int(dane[i+3]) and int(dane[i+3]) < int(dane[i+4]) and int(dane[i+4]) < int(dane[i+5]):
        odp3 += 1
    if int(dane[i]) > int(dane[i+1]) and int(dane[i+1]) < int(dane[i+2]) and int(dane[i+2]) < int(dane[i+3]) and int(dane[i+3]) < int(dane[i+4]) and int(dane[i+4]) < int(dane[i+5]):
        odp3 +=1

print(odp3)
```

Rysunek 21. Fragment rozwiązania zadania 3.3. bez uwzględnienia wszystkich warunków.

Rysunki 22. i 23. przedstawiają rozwiązanie poprawne. Zdefiniowano funkcję sprawdzającą czy dany ciąg 6-elementowy jest rosnąco-malejący, a następnie użyto jej do rozwiązania zadania. W tym przypadku nie wystąpiła konieczność porównywania elementów środkowych dla ciągu „trzy i trzy”, ponieważ niezależnie od wyniku porównania tych elementów funkcja zwróci 1 i zakończy działanie, więc takie ciągi nie będą zliczane powtórnie.

```
7   int czy_rosnaco_malejacy(int a[6]){
8       //dwa i cztery:
9       int prawda1 = 0 , prawda2 = 0;
10      if(a[0] <a[1])
11          prawda1 = 1;
12      if(a[2]>a[3] && a[3]>a[4]&&a[4]>a[5])
13          prawda2 =1;
14      if(prawda1 == 1 && prawda2 == 1)
15          return 1;
16      //trzy i trzy:
17      prawda1 = 0;
18      prawda2 = 0;
19      if(a[0] <a[1]&&a[1]<a[2])
20          prawda1 = 1;
21      if(a[3]>a[4]&&a[4]>a[5])
22          prawda2 =1;
23      if(prawda1 == 1 && prawda2 == 1)
24          return 1;
25      //cztery i dwa:
26      prawda1 = 0;
27      prawda2 = 0;
28      if(a[0] <a[1]&&a[1]<a[2]&&a[2]<a[3])
29          prawda1 = 1;
30      if(a[4]>a[5])
31          prawda2 =1;
32      if(prawda1 == 1 && prawda2 == 1)
33          return 1;
34      return 0;
35  }
```

Rysunek 22. funkcja sprawdzająca czy ciąg jest rosnąco-malejący.

```
68      int ciag[6], ile3 = 0;;
69      for(int i = 0; i<9995; i++){
70          ciag[0] = pi[i];
71          ciag[1] = pi[i+1];
72          ciag[2] = pi[i+2];
73          ciag[3] = pi[i+3];
74          ciag[4] = pi[i+4];
75          ciag[5] = pi[i+5];
76          if(czy_rosnaco_malejacy(ciag) == 1){
77              ile3++;
78
79          }
80      }
81      cout<<ile3<<endl;
```

Rysunek 23. zastosowanie funkcji sprawdzającej ciąg 6-elementowy w programie.

Na rysunku 24. przedstawiono poprawne rozwiązanie w języku Python z użyciem funkcji sprawdzającej, czy ciąg jest rosnąco-malejący wg definicji z zadania.

```python
def rosnacy(k,c):
    for i in range(k - 1):
        if int(c[i]) >= int(c[i+1]):
            return False
    return True
def malejacy(k,c):
    for i in range(k, 5):
        if int(c[i]) <= int(c[i+1]):
            return False
    return True

def rosnącomalejący(c):
    for k in range(2,5):
        if rosnący(k,c) == True and malejący(k,c) == True:
            return True
    return False

fragmenty = []
zad3 = 0
for i in range(9995):
    fragmenty.append(dane[i]+dane[i+1]+dane[i+2] + dane[i+3] + dane[i+4] + dane[i+5])
    if rosnącomalejący(fragmenty[i]) == True:
        zad3 += 1

```

Rysunek 24. rozwiązanie zadania 3.3. w języku Python.

W zadaniu 3.4. należało znaleźć najdłuższy ciąg rosnąco-malejący złożony z kolejnych cyfr zapisanych w pliku pi.txt. W odpowiedzi należało podać numer wiersza, od którego zaczyna się najdłuższy ciąg, oraz należało wypisać cyfry należące do tego ciągu. Po analizie poprzedniego zadania można zauważyć, że prawdopodobnie najwygodniejsza będzie strategia oparta o drugi sposób rozwiązania zadania 3.3. – czyli zaczynając od aktualnej cyfry, sprawdzamy kolejne tak długo, jak długo kolejne cyfry są większe, czyli do końca ciągu rosnącego, a następnie sprawdzamy, czy kolejne cyfry tworzą ciąg malejący i także przechodzimy do końca tego ciągu – jednocześnie zliczając liczbę cyfr. Jeśli znaleziony ciąg jest dłuższy niż poprzednio zapamiętany, to zapamiętujemy go (wiersz, od którego się zaczął oraz kolejne cyfry ciągu).

Rysunki 25. i 26. przedstawiają przykładowe poprawne rozwiązania zapisane w języku Python i C++.

Wielu zdających nie podejmowało próby rozwiązania zadań 3.3. i 3.4.

```python
for i in range(len(dane)-1):
    poz, ciag = i+1, dane[i]
    ros, mal = True, False
    ileros, ilemal = 1, 0
    for j in range(i+1, len(dane)):
        if ros and not mal:
            if dane[j] > dane[j-1]:
                ciag += dane[j]
                ileros += 1
            elif ileros >= 2:
                ciag += dane[j]
                ros = False
                mal = True
            else:
                break
        elif not ros and mal:
            if dane[j] < dane[j-1]:
                ciag += dane[j]
                ilemal += 1
            elif ilemal >= 2:
                if len(ciag) > len(maksciag):
                    maksciag = ciag
                    makspoz = poz
                break
            else:
                break
        else:
            break

print(f'Pozycja: {makspoz}')
print(f'Ciąg: {maksciag}')
```

Rysunek 25. Przykładowe rozwiązanie zadania 3.4. w języku Python.

```cpp
int maxLeng=0;
int maxIndeks=0;
for(int i=0;i<p;i++)
{
    k=1;
    l=i+1;
    while(l<p&&input[l]>input[l-1])
    {
        l++;
        k++;
    }
    m=1;
    l++;
    while(l<p&&input[l]<input[l-1])
    {
        l++;
        m++;
    }
    if(m+k>maxLeng)
        {
            maxLeng=m+k;
            maxIndeks=i;
        }
}
cout<<maxIndeks+1<<endl; //numeruje od zera dlatego musze dodać jeden :(
for(int i=maxIndeks;i<maxIndeks+maxLeng;i++)
{
    cout<<input[i];
}
cout<<endl;
```

Rysunek 26. Przykładowe rozwiązanie zadania 3.4. w języku C++.

Wnioski i rekomendacje

Średni uzyskany przez absolwentów liceów wynik z egzaminu maturalnego z informatyki jest porównywalny z wynikami absolwentów liceów w ubiegłym roku. Podobnie jak w poprzednich latach, najwięcej trudności sprawiły zdającym rozwiązania zadań wymagające zapisania algorytmu i napisania programu w języku programowania, a także zadania symulacyjne.

Nowa formuła egzaminu po raz pierwszy zagwarantowała zdającym możliwość korzystania z komputera podczas całego egzaminu maturalnego z informatyki. Umożliwiało to użycie komputera także w przypadku zadań, których rozwiązania zapisuje się tylko w arkuszu egzaminacyjnym.

Z analizy uzyskanych przez tegorocznych absolwentów liceów wyników można wnioskować, że problematyczne dla zdających okazało się przede wszystkim zaplanowanie kolejnych kroków rozwiązania, które było kluczowe dla poprawnego zapisu algorytmu zarówno w postaci pseudokodu, jak również w języku programowania podczas rozwiązywania zadań praktycznych, wymagających nie tylko udzielenia prawidłowej odpowiedzi, ale również przedstawienia do oceny komputerowej realizacji zadania. Zdarzały się także przypadki braku załączenia plików z komputerową realizacją, co w tego typu przypadkach skutkowało zerową oceną danego zadania. Zdający lepiej radzili sobie z zadaniami, które nie wymagały głębszej analizy problemu, a poprawne rozwiązanie było możliwe do uzyskania już po wykonaniu niewielkiej liczby operacji i z uwzględnieniem małej ilości warunków. Świadczy to o rozwiniętych umiejętnościach posługiwania się narzędziami komputerowymi do rozwiązywania postawionych w zadaniach problemów i jednocześnie wskazuje na niskie umiejętności w obszarze układania i programowania algorytmów.

W związku z powyższymi wnioskami rekomenduje się, aby podczas przygotowywania uczniów do egzaminu maturalnego z informatyki na poziomie rozszerzonym:

1) zwracać uwagę na odpowiedni sposób zaplanowania czasu przeznaczonego na rozwiązywanie poszczególnych zadań podczas egzaminu. W poprzednich latach egzamin składał się z dwóch części, co uniemożliwiało zdającym dowolny podział czasu przeznaczonego na różne typy zadań – obecnie możliwe jest bardziej elastyczne rozplanowanie czasu, ale jednocześnie wymaga to od zdających pewnej umiejętności planowania jego podziału podczas egzaminu,
2) kształcić umiejętność rozwiazywania zadań praktycznych, stanowiących większość wśród zadań ujętych w arkuszu egzaminacyjnym z informatyki, w szczególności poprzez:
   - ćwiczenia polegające na rozwiązywaniu typowych zadań praktycznych, tak by nie tracić czasu na rozwiązywanie zadań niewymagających dużej inwencji,
   - zwracanie uwagi na konieczność czytelnego wskazania komputerowej realizacji zadania, np.: wpisywanie nazw plików w przeznaczonym do tego miejscu, umieszczanie rozwiązania każdego zadania w odrębnej, odpowiednio nazwanej zakładce arkusza kalkulacyjnego, zapisywanie kwerend pod odpowiednimi nazwami itp.,
   - dokładną analizę treści zadań oraz dobór odpowiednich narzędzi do ich rozwiązywania,
   - omawianie z uczniami popełnianych błędów i wskazywanie przyczyn tych błędów

3) kształcić umiejętność zapisu algorytmu w postaci pseudokodu lub językach programowania z uwzględnieniem treści i warunków zadania. Każdorazowo kłaść nacisk na zgodność stosowanych w zapisie algorytmu operacji z wymienionymi w treści zadania i nieużywanie dostępnych w językach programowania wbudowanych funkcji oraz operatorów innych niż wymienione.