Dorota Roman-Jurdzińska

INFORMATYKA

1. Część informacyjna

Do egzaminu maturalnego z informatyki, który się odbył się 22 maja 2013 r., przystąpiło po raz pierwszy **263** absolwentów szkół ponadgimnazjalnych, w tym **144** osób zdających wybrało ten przedmiot na poziomie podstawowym, zaś **119** na poziomie rozszerzonym (tabela 1).

Tabela 1. Liczby uczniów na egzaminie maturalnym z informatyki – zestaw standardowy

Zdający	Liczba egzaminów		
	Ogółem	poziom podstawowy	poziom rozszerzony
Okręg			
LO	138	46	92
LP	1		1
T	123	97	26
LU	1	1	
TU			
RAZEM	263	144	119
województwo dolnośląskie			
LO	103	31	72
LP	1		1
T	95	70	25
LU	1	1	
TU			
RAZEM	200	102	98
województwo opolskie			
LO	35	15	20
LP			
T	28	27	1
LU			
TU			
RAZEM	63	42	21

Średni wynik procentowy z informatyki zdawanej na poziomie podstawowym wyniósł **42,9%** (w ubiegłym roku 34,7%). Dla poziomu rozszerzonego średni wynik procentowy wyniósł **59%** i jest taki sam jak w roku ubiegłym (tabela 2).

Tabela 2. Średni wynik procentowy zdających egzamin maturalny z informatyki

Typ szkoły	Średni wynik procentowy	
	poziom podstawowy	poziom rozszerzony
OKE Wrocław		
LO	46,6	63,5
LP		30,0
T	41,5	44,1
LU	12,0	
TU		
RAZEM	42,9	59,0
województwo dolnośląskie		
LO	46,8	66,4
LP		30,0
T	38,3	43,9
LU	12,0	
TU		
RAZEM	40,6	60,3
województwo opolskie		
LO	46,3	53,1
LP		
T	49,9	48,0
LU		
TU		
RAZEM	48,6	52,9

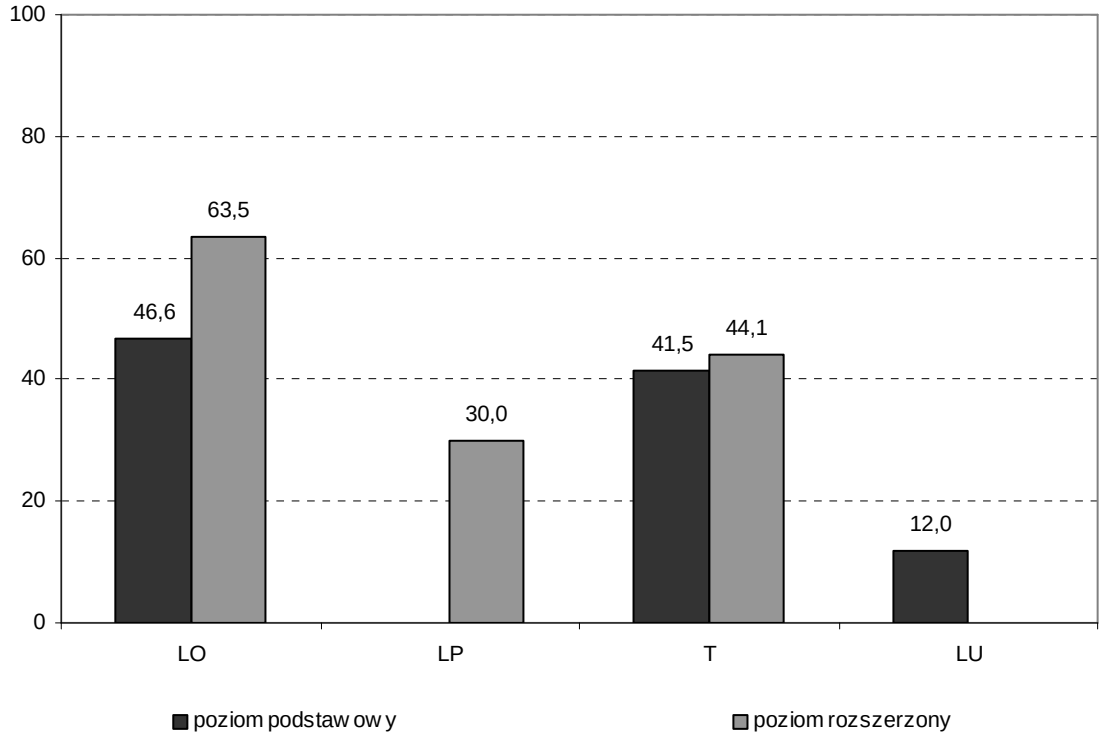


Diagram 1. Średnie wyniki procentowe w różnych typach szkół

Wybrane dane statystyczne

Poziom podstawowy egzaminu

Wskaźnik łatwości arkusza egzaminacyjnego jest określony przez średni wynik procentowy uzyskany przez zdających. W naszym okręgu średni wynik procentowy uzyskany przez zdających na egzaminie maturalnym z informatyki na poziomie podstawowym jest równy 42,9%.

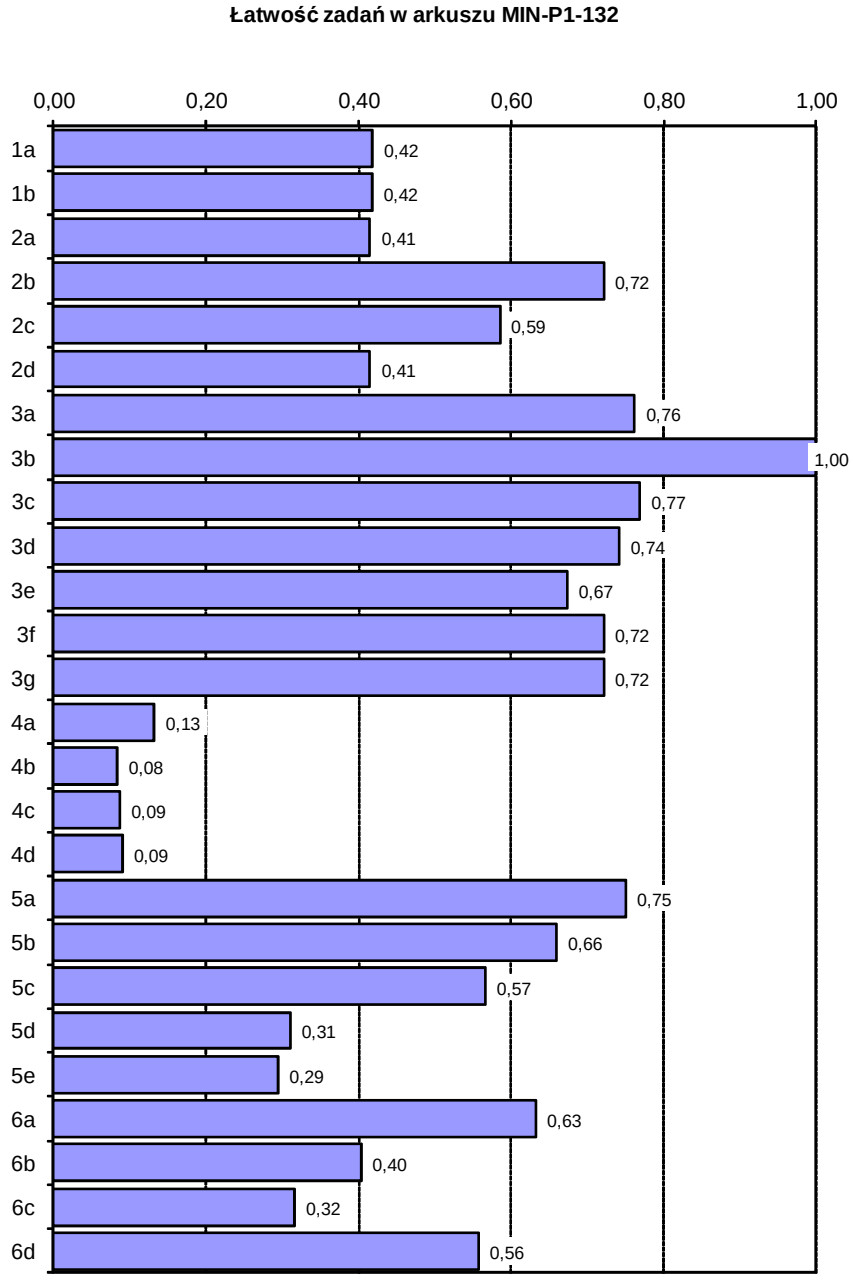


Diagram 2. Łatwość zadań – poziom podstawowy

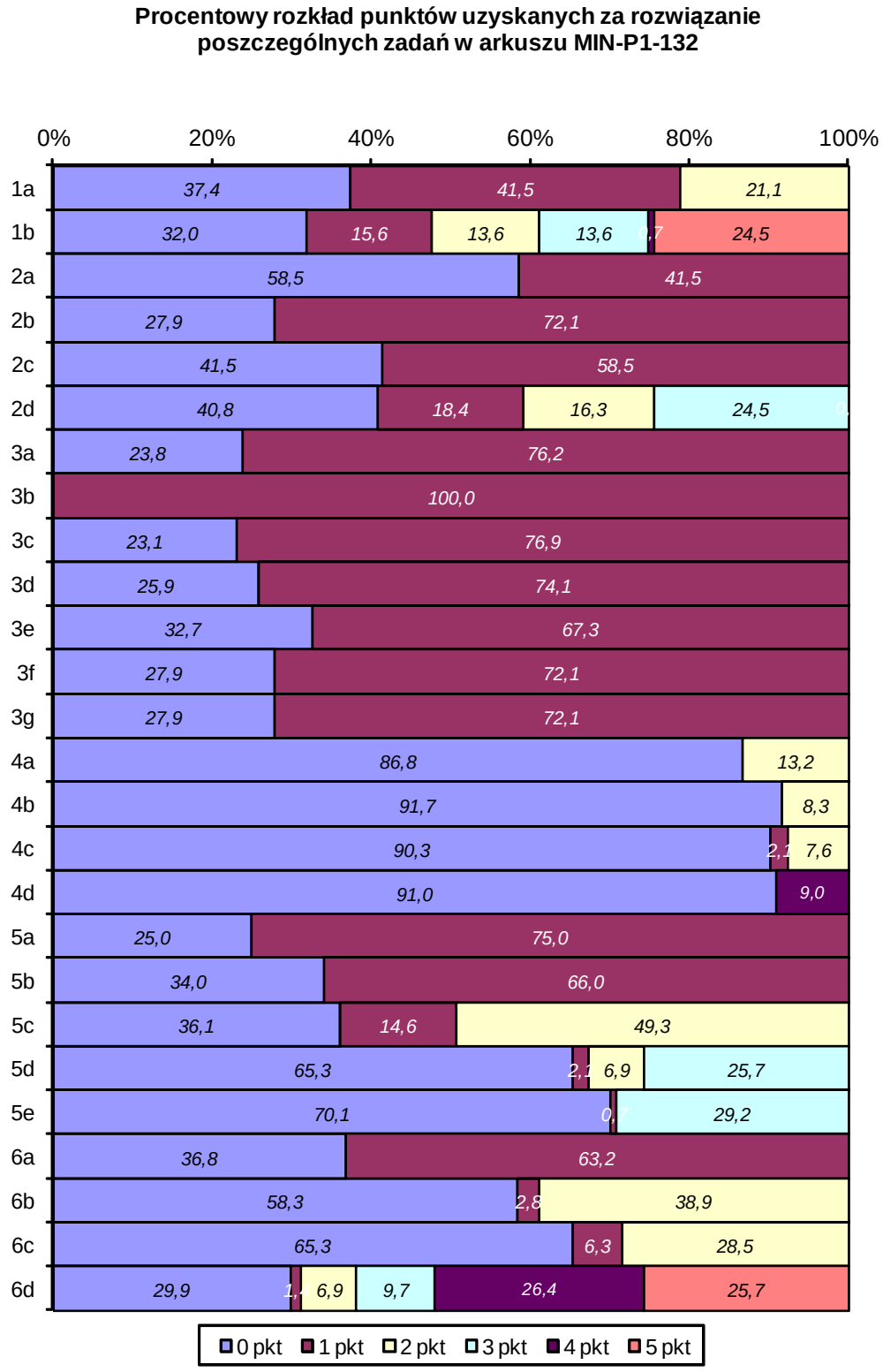


Diagram 3. Procentowy rozkład punktów za rozwiązanie poszczególnych zadań – poziom podstawowy

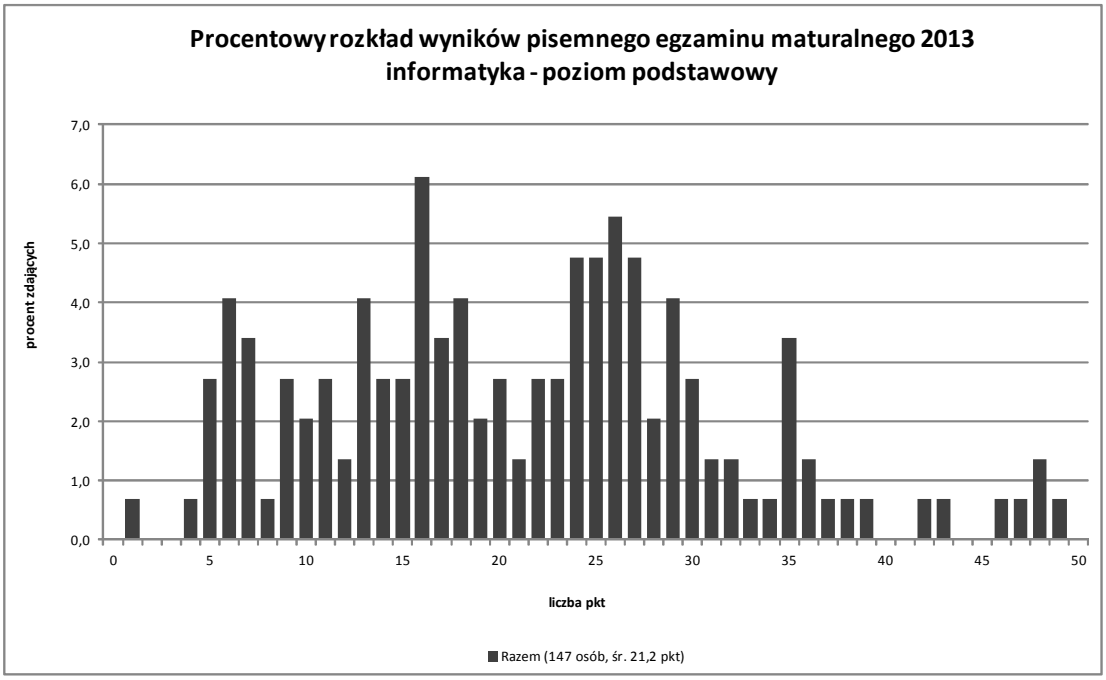


Diagram 4. Procentowy rozkład wyników– poziom podstawowy

Poziom rozszerzony egzaminu

W naszym okręgu średni wynik procentowy uzyskany przez zdających na egzaminie maturalnym z informatyki na poziomie rozszerzonym jest równy 59,0%.

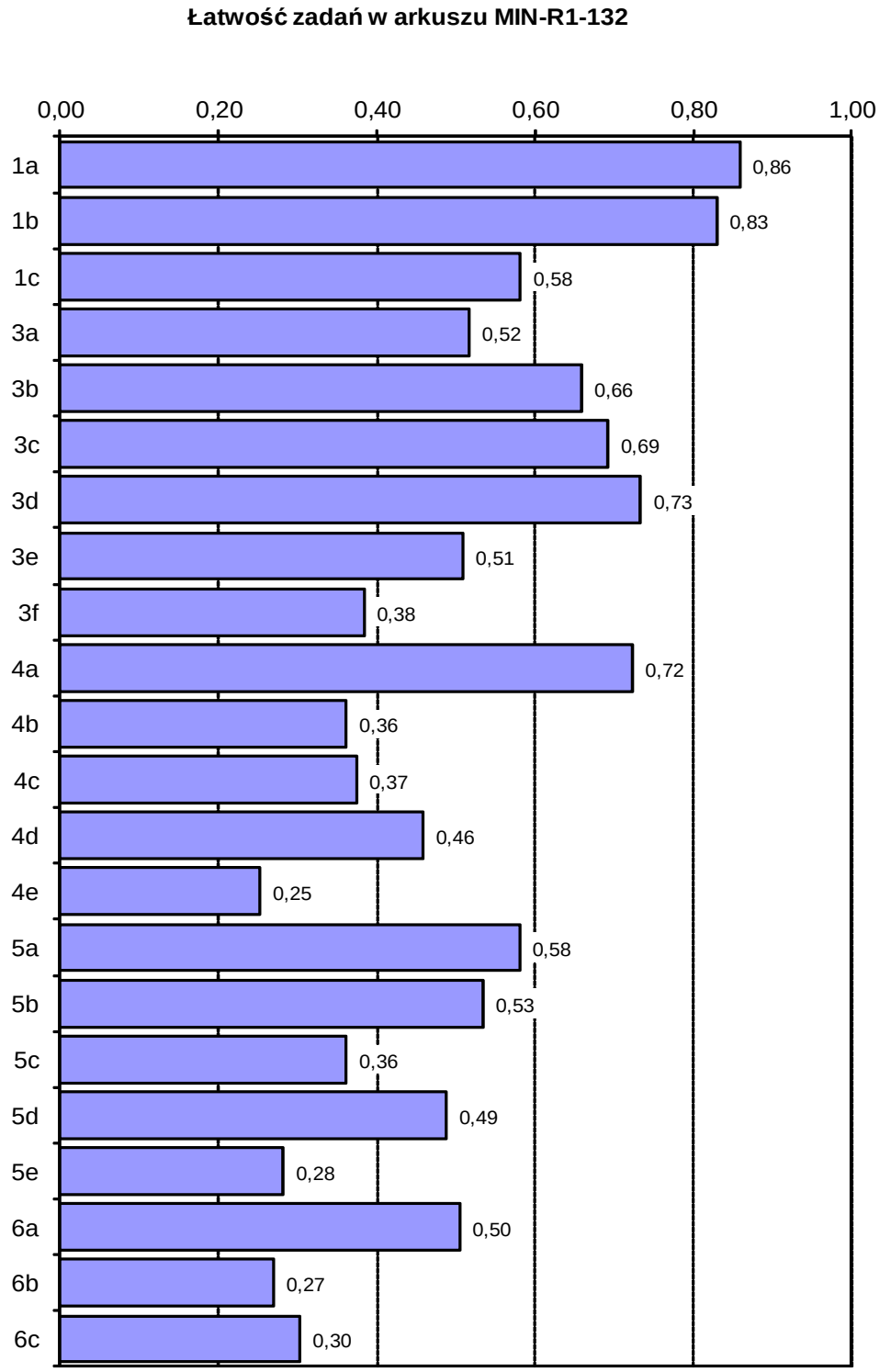


Diagram 5. Łatwość zadań – poziom rozszerzony

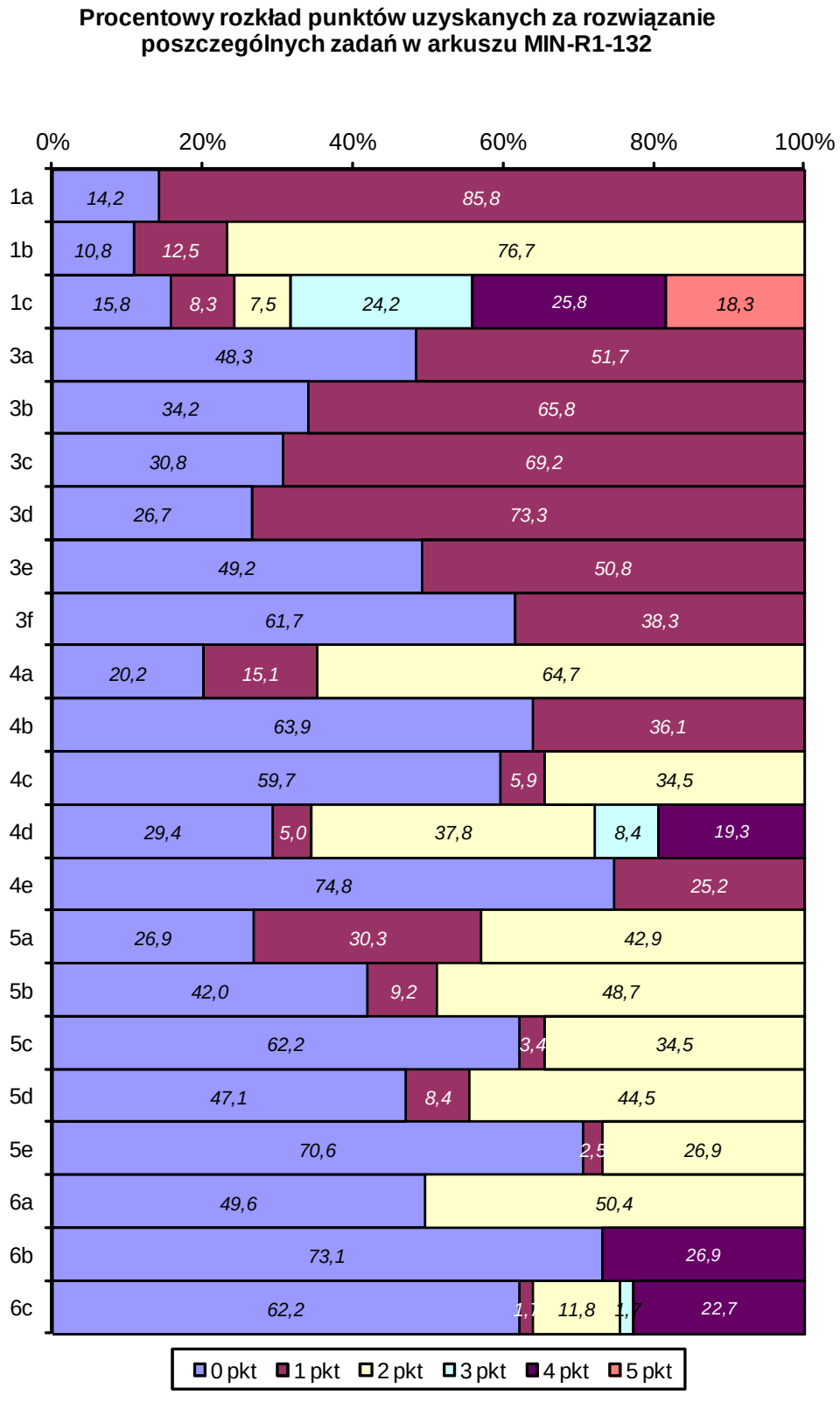


Diagram 6. Procentowy rozkład punktów za rozwiązanie poszczególnych zadań – poziom rozszerzony

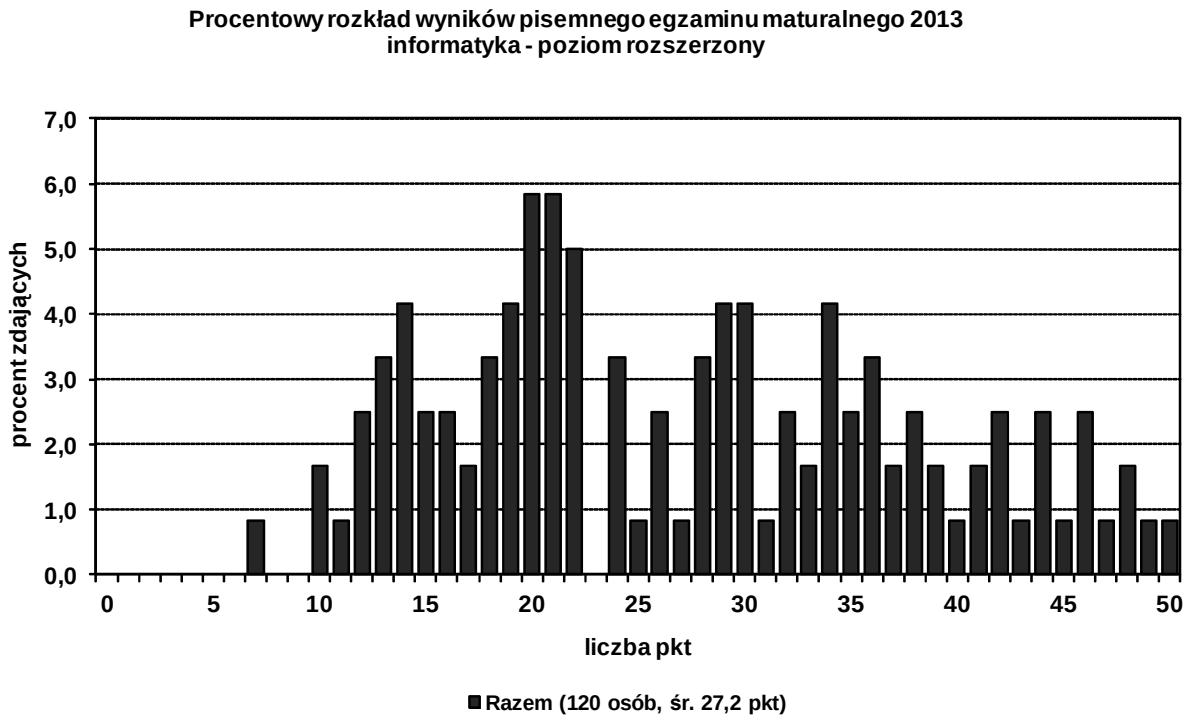


Diagram 7. Procentowy rozkład wyników– poziom rozszerzony

Opis arkuszy egzaminacyjnych

Centralna Komisja Egzaminacyjna przygotowała dwa zestawy egzaminacyjne. Jeden dla poziomu podstawowego, drugi dla poziomu rozszerzonego. Każdy zestaw składał się z dwóch arkuszy (część teoretyczna i praktyczna), z których każdy zawierał 3 zadania o zróżnicowanej punktacji i wymagające samodzielnego sformułowania odpowiedzi. Za poprawne rozwiązanie wszystkich zadań z obu części zdający mógł uzyskać 50 punktów na każdym poziomie.

Arkusze egzaminacyjne wraz z kluczami punktowania są dostępne na stronie internetowej OKE we Wrocławiu (www.oke.wroc.pl) oraz CKE (www.cke.edu.pl).

Poziom podstawowy

Tabela 3. Łatwość zadań na poziomie podstawowym

Nr zad.	Obszar standardów	Sprawdzana umiejętność	Łatwość zadania
1.a	Korzystanie z informacji	Analiza liczby wykonywanych w algorytmie operacji (II.5).	0,42
1.b	Wiadomości i rozumienie	Znajomość podstawowych technik algorytmicznych i algorytmów (I.7).	0,42
2.a	Korzystanie z informacji	Analiza liczby wykonywanych w algorytmie operacji (II.5).	0,41
2.b	Korzystanie z informacji	Zastosowanie podstawowych algorytmów w rozwiązywaniu problemów informatycznych (II.5).	0,72
2.c	Korzystanie z informacji	Analiza liczby wykonywanych w algorytmie operacji (II.5).	0,59
2.d	Korzystanie z informacji	Zastosowanie podstawowych algorytmów w rozwiązywaniu problemów informatycznych (II.5).	0,41

3.a	Wiadomości i rozumienie	Znajomość podstawowych pojęć związanych z relacyjnymi bazami danych (I.10).	0,76
3.b	Wiadomości i rozumienie	Znajomość typowych narzędzi informatycznych i ich zastosowań (I.3).	1,00
3.c	Wiadomości i rozumienie	Znajomość technik algorytmicznych i algorytmów (I.7).	0,77
3.d	Korzystanie z informacji	Wykonywanie obliczeń przy pomocy wbudowanych funkcji i zaprojektowanych formuł (II.1).	0,74
3.e	Wiadomości i rozumienie	Znajomość sposobów reprezentowania informacji w komputerze (I.6).	0,67
3.f	Wiadomości i rozumienie	Znajomość podstawowej terminologii związanej z sieciami komputerowymi: rodzaje sieci, protokoły, podstawowe usługi sieciowe i sposoby ochrony zasobów (I.3).	0,72
3.g	Wiadomości i rozumienie	Znajomość zasad etycznych i prawnych związanych z wykorzystywaniem informacji i oprogramowania (I.11).	0,72
4.a	Korzystanie z informacji	Posłużenie się kompilatorem wybranego języka programowania (II.2).	0,13
	Tworzenie informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (III.2).	
4.b	Korzystanie z informacji	Posłużenie się kompilatorem wybranego języka programowania (II.2).	0,08
	Tworzenie informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (III.2).	
4.c	Korzystanie z informacji	Posłużenie się kompilatorem wybranego języka programowania (II.2)	0,09
	Tworzenie informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (III.2)	
4.d	Korzystanie z informacji	Posłużenie się kompilatorem wybranego języka programowania (II.2)	0,09
	Tworzenie informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (III.2)	
5.a	Korzystanie z informacji	Zastosowanie metod wyszukiwania i przetwarzania informacji w relacyjnych bazach danych (II.4)	0,75
	Tworzenie informacji	Zaprojektowanie relacyjnej bazy danych i wykorzystanie do jej realizacji systemu bazy danych (III.3)	
5.b	Korzystanie z informacji	Zastosowanie metod wyszukiwania i przetwarzania informacji w relacyjnych bazach danych (II.4)	0,66
	Tworzenie informacji	Zaprojektowanie relacyjnej bazy danych i wykorzystanie do jej realizacji systemu bazy danych (III.3)	
5.c	Korzystanie z informacji	Zastosowanie metod wyszukiwania i przetwarzania informacji w relacyjnych bazach danych (II.4)	0,57
	Tworzenie informacji	Zaprojektowanie relacyjnej bazy danych i wykorzystanie do jej realizacji systemu bazy danych (III.3)	
5.d	Korzystanie z informacji	Zastosowanie metod wyszukiwania i przetwarzania informacji w relacyjnych bazach danych (II.4)	0,31
	Tworzenie informacji	Zaprojektowanie relacyjnej bazy danych i wykorzystanie do jej realizacji systemu bazy danych (III.3)	

5.e	Korzystanie z informacji	Zastosowanie metod wyszukiwania i przetwarzania informacji w relacyjnych bazach danych (II.4)	0,29
	Tworzenie informacji	Zaprojektowanie relacyjnej bazy danych i wykorzystanie do jej realizacji systemu bazy danych (III.3)	
6.a	Korzystanie z informacji	Dobranie właściwego programu (użytkowego lub własnoręcznie napisanego) do rozwiązywanego zadania (II.6) Zastosowanie odpowiedniego formatowania danych i tabeli oraz wykonanie obliczeń przy pomocy wbudowanych oraz zaprojektowanych formuł (II.1)	0,63
6.b	Korzystanie z informacji	Dobranie właściwego programu (użytkowego lub własnoręcznie napisanego) do rozwiązywanego zadania (II.6) Zastosowanie odpowiedniego formatowania danych i tabeli oraz wykonanie obliczeń przy pomocy wbudowanych oraz zaprojektowanych formuł (II.1)	0,40
6.c	Korzystanie z informacji	Dobranie właściwego programu (użytkowego lub własnoręcznie napisanego) do rozwiązywanego zadania (II.6) Zastosowanie odpowiedniego formatowania danych i tabeli oraz wykonanie obliczeń przy pomocy wbudowanych oraz zaprojektowanych formuł (II.1)	0,32
6.d	Korzystanie z informacji	Dobranie właściwego programu (użytkowego lub własnoręcznie napisanego) do rozwiązywanego zadania (II.6) Zastosowanie odpowiedniego formatowania danych i tabeli oraz wykonanie obliczeń przy pomocy wbudowanych oraz zaprojektowanych formuł (II.1) Posłużenie się arkuszem kalkulacyjnym w celu graficznego zobrazowania informacji adekwatnie do ich charakteru (II.1)	0,56

Poziom rozszerzony

Tabela 4. Łatwość zadań na poziomie rozszerzonym

Nr zad.	Obszar standardów	Sprawdzana umiejętność	Łatwość zadania
1.a	Wiadomości i rozumienie	Znajomość systemów liczbowych mających zastosowanie w informatyce (I.3).	0,86
1.b	Wiadomości i rozumienie	Znajomość systemów liczbowych mających zastosowanie w informatyce (I.3).	0,83
1.c	Wiadomości i rozumienie	Znajomość podstawowych technik algorytmicznych i algorytmów (I.4).	0,58
3.a	Wiadomości i rozumienie	Znajomość podstawowej terminologii związanej z sieciami komputerowymi: rodzaje sieci, protokoły (I.4 PP).	0,52
3.b	Wiadomości i rozumienie	Umiejętność analizy działania algorytmu dla wskazanych danych (I.7PP).	0,66
3.c	Wiadomości i rozumienie	Znajomość zasad konwersji liczb pomiędzy różnymi systemami pozycyjnymi (I.3).	0,69
3.d	Wiadomości i rozumienie	Znajomość zasad etycznych i prawnych związanych z wykorzystywaniem oprogramowania (I.11PP).	0,73
3.e	Wiadomości i rozumienie	Znajomość podstawowej terminologii związanej z sieciami komputerowymi (I.4.PP).	0,51
3.f	Wiadomości i rozumienie	Umiejętność analizy algorytmu dla wskazanych danych (I.7PP).	0,38

4.a	Korzystanie z informacji	Modelowanie zjawisk i procesów z różnych dziedzin życia (II.3).	0,72
	Tworzenie informacji	Wykorzystywanie metod informatyki do rozwiązywania problemów (III.2).	
4.b	Korzystanie z informacji	Modelowanie zjawisk i procesów z różnych dziedzin życia (II.3).	0,36
	Tworzenie informacji	Wykorzystywanie metod informatyki do rozwiązywania problemów (III.2).	
4.c	Korzystanie z informacji	Modelowanie zjawisk i procesów z różnych dziedzin życia (II.3).	0,37
	Tworzenie informacji	Wykorzystywanie metod informatyki do rozwiązywania problemów (III.2).	
4.d	Tworzenie informacji	Wykonywanie obliczeń przy pomocy wbudowanych funkcji i zaprojektowanych formuł, graficzne obrazowanie informacji adekwatnie do jej charakteru (II.1.PP).	0,46
4.e	Tworzenie informacji	Modelowanie zjawisk i procesów z różnych dziedzin życia (II.3).	0,25
5.a	Tworzenie informacji	Analiza problemu i zbioru danych, którego rozwiązanie wymaga zaprojektowanie i utworzenie relacyjnej bazy danych(tabeli i relacji między nimi) z uwzględnieniem zawartych informacji (III.3).	0,58
	Korzystanie z informacji	Wyszukanie informacji w bazie danych stosując różne techniki (w tym zapytania) oraz zastosowanie metod optymalizujących wyszukiwanie (indeksowanie) (II.1).	
5.b	Tworzenie informacji	Analiza problemu i zbioru danych, którego rozwiązanie wymaga zaprojektowanie i utworzenie relacyjnej bazy danych(tabeli i relacji między nimi) z uwzględnieniem zawartych informacji (III.3).	0,53
	Korzystanie z informacji	Wyszukanie informacji w bazie danych stosując różne techniki (w tym zapytania) oraz zastosowanie metod optymalizujących wyszukiwanie (indeksowanie) (II.1).	
5.c	Tworzenie informacji	Analiza problemu i zbioru danych, którego rozwiązanie wymaga zaprojektowanie i utworzenie relacyjnej bazy danych(tabeli i relacji między nimi) z uwzględnieniem zawartych informacji (III.3).	0,36
	Korzystanie z informacji	Wyszukanie informacji w bazie danych stosując różne techniki (w tym zapytania) oraz zastosowanie metod optymalizujących wyszukiwanie (indeksowanie) (II.1).	
5.d	Tworzenie informacji	Analiza problemu i zbioru danych, którego rozwiązanie wymaga zaprojektowanie i utworzenie relacyjnej bazy danych(tabeli i relacji między nimi) z uwzględnieniem zawartych informacji (III.3).	0,49
	Korzystanie z informacji	Wyszukanie informacji w bazie danych stosując różne techniki (w tym zapytania) oraz zastosowanie metod optymalizujących wyszukiwanie (indeksowanie) (II.1).	
5.e	Tworzenie informacji	Analiza problemu i zbioru danych, którego rozwiązanie wymaga zaprojektowanie i utworzenie relacyjnej bazy danych(tabeli i relacji między nimi) z uwzględnieniem zawartych informacji (III.3).	0,28
	Korzystanie z informacji	Wyszukanie informacji w bazie danych stosując różne techniki (w tym zapytania) oraz zastosowanie metod optymalizujących wyszukiwanie (indeksowanie) (II.1).	
6.a	Wiadomości i rozumienie	Znajomość technik algorytmicznych i algorytmów (I.4)	0,50

	Korzystanie z informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (policzenie liczb spełniających warunek określony w zadaniu) (III.2)	
6.b	Wiadomości i rozumienie	Znajomość systemów liczbowych mających zastosowanie w informatyce oraz zasad konwersji pomiędzy różnymi systemami pozycyjnymi (I.3)	0,27
	Korzystanie z informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (konwersja pomiędzy pozycyjnymi systemami liczbowymi: ósemkowym i dziesiętnym) (III.2)	
6.c	Wiadomości i rozumienie	Znajomość technik algorytmicznych i algorytmów (I.4)	0,30
	Korzystanie z informacji	Sformułowanie informatycznego rozwiązania problemu przez dobór algorytmu i odpowiednich typów oraz struktur danych i zaimplementowanie go w wybranym języku programowania (III.2).	

2. Rozwiążmy zadania

Egzamin maturalny z informatyki, zarówno na poziomie podstawowym jak i rozszerzonym, składa się z dwóch części. Część pierwsza polega na rozwiązywaniu zadań bez korzystania z komputera. W części drugiej zdający rozwiązuje zadania przy użyciu komputera i wybranych wcześniej przez siebie narzędzi (język programowania, programy użytkowe). Egzamin jest egzaminem pisemnym, sprawdzającym wiadomości i umiejętności określone w Standardach wymagań egzaminacyjnych.

Jednym z trzech obszarów obejmujących standardy wymagań egzaminacyjnych są wiadomości i rozumienie. Zdający powinien znać i rozumieć podstawowe pojęcia i metody związane z informatyką. Na poziomie rozszerzonym jest to między innymi znajomość systemów liczbowych, podstawowych własności algorytmów (pojęcie algorytmu, różne sposoby jego zapisu, zgodność algorytmu ze specyfikacją) oraz różnych technik algorytmicznych i algorytmów.

W obszarze dotyczącym korzystania z informacji zdający powinien zastosować posiadaną wiedzę do rozwiązywania zadań zarówno teoretycznych, jak i praktycznych. W zakresie algorytmiki zdający powinien stosować podstawowe algorytmy i struktury danych w rozwiązywaniu problemów informatycznych oraz wykonywać kolejne etapy prowadzące do otrzymania rozwiązania zadania, od sformułowania specyfikacji problemu do zapisu właściwej części algorytmu w jednej z wybranych notacji, natomiast w części praktycznej – m.in. posługiwać się typowymi programami użytkowymi, dobierać właściwy program (użytkowy lub napisany własnoręcznie) do rozwiązania zadania oraz stosować metody wyszukiwania i przetwarzania informacji w relacyjnych bazach danych.

W obszarze dotyczącym tworzenia informacji zdający powinien stosować metody informatyczne do rozwiązywania problemów, a w szczególności określać sytuacje problemową, definiować problem, tworzyć specyfikację do postawionego zadania oraz proponować i analizować przedstawione rozwiązanie. Sformułowane rozwiązanie powinien realizować przy pomocy różnych narzędzi informatycznych, np. w wybranym języku programowania poprzez dobór odpowiedniego algorytmu. Ponadto zdający projektuje relacyjne bazy danych.

W pierwszej części tegorocznego arkusza egzaminacyjnego na poziomie rozszerzonym występują 2 zadania związane z algorytmami: analiza algorytmu oraz ułożenie i zapisanie algorytmu (zadanie drugie zostało unieważnione z powodu błędów w treści zadania). Trzecie zdanie jest zadaniem testowym obejmującym ogólną wiedzę informatyczną.

W części drugiej egzaminu maturalnego z informatyki zdający rozwiązywał zadania przy użyciu komputera oraz wybranych wcześniej przez siebie narzędzi (są to język programowania oraz programy użytkowe MsOffice lub OpenOffice). W części tej można wyodrębnić następujące typy zadań: programistyczne, bazodanowe, obliczeniowo-symulacyjne.

Zadanie 1 z pierwszej części arkusza dotyczy sposobu zapisywania liczb w kodzie U1. W treści zadania wyjaśniono, na czym polega zapis liczb całkowitych tym sposobem. Podpunkty a) i b) sprawdzają zrozumienie notacji U1. W podpunkcie a) zadania wystarczy zamienić liczby zapisane dziesiętne na liczby zapisane w kodzie U1, a w podpunkcie b) trzeba zamienić liczby binarne zapisane w kodzie U1 na liczby zapisane w systemie dziesiętnym.

Dopiero podpunkt c) wymaga napisana algorytmu obliczania wartości liczby zapisanej w kodzie U1. To zadanie można rozwiązać na wiele sposobów, poniżej przedstawimy jeden z nich.

Przeanalizujmy, czego wymaga od nas zamiana liczby z kodu U1 na system dziesiętny. Przede wszystkim musimy sprawdzić, czy liczba jest dodatnia, czy ujemna, czyli sprawdzamy pierwszy bit zapisu. Jeśli liczba jest ujemna, tzn. gdy `bin[1]=1`, to negujemy pozostałe bity liczby, od drugiego do ostatniego. Następnie zamieniamy liczbę z systemu binarnego na dziesiętny, korzystając np. ze schematu Hornera. Ostatnim krokiem jest ustalenie znaku liczby dziesiętnej na podstawi wartości wskazanego powyżej pierwszego bitu. Poniżej przedstawiamy przykładowy fragment algorytmu z komentarzem:

```
x ← 0  // liczba dziesiętna
i ← 2  // pętlę zaczynamy od drugiego bitu
jeżeli bin[1] = 1 to    // jeśli liczba jest ujemna,
 dopóki i ≤ d    // to negujemy bity
      jeżeli bin[i] = 1 to bin[i] ← 0
      w przeciwnym wypadku bin[i] ← 1
      i ← i+1
i←2  // pętlę zaczynamy od drugiego bitu
dopóki i ≤ d     // zamieniamy liczbę binarną na system
dziesiętny
 x ← x*2 + bin[i]
 i ← i+1
jeżeli bin[1] = 1 to x ← x*(-1)   // ustalamy znak
liczby dziesiętnej
```

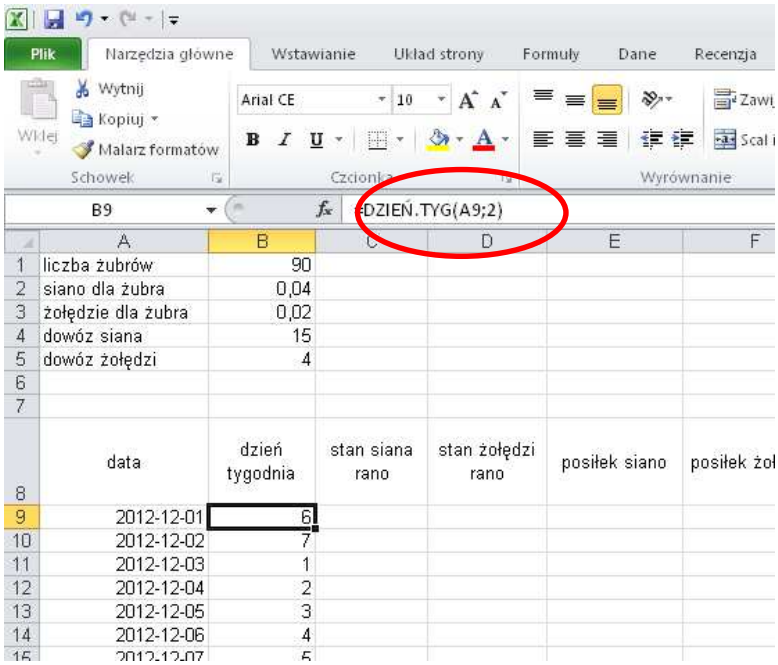
Pierwsze zadanie z drugiej praktycznej części arkusza jest zadaniem symulacyjnym; można je rozwiązać przy pomocy arkusza kalkulacyjnego lub pisząc program w wybranym przez siebie języku programowania. Jego najtrudniejszą częścią jest prawidłowe symulowanie kolejnych zdarzeń, które występują cyklicznie, z pewną częstotliwością i są od siebie zależne.

Rozwiązanie rozpoczynamy od wpisana stałych elementów zadania do arkusza tzn. liczby żubrów, ilości siana oraz żołędzi przypadających na żubra podczas jednego posiłku. Następnie wpisujemy okres symulacji: od 1 grudnia 2012 r. do 28 lutego 2013 r. Ponieważ w zadaniu pojawiają się dni tygodnia, dobrze jest ustalić jedną kolumnę, która będzie przechowywać te dni.

W każdym dniu symulacji zachodzi wiele możliwych zdarzeń (np. dostawy siana lub żołędzi, posiłki żubrów), wskazane jest rozpisanie tego, co dzieje się podczas całego dnia. Tutaj bardzo istotne jest przeczytanie najpierw całej treści zadania, żeby wiedzieć, jakie wydarzenia będą potrzebne oraz jakie elementy będą przydatne przy udzielaniu odpowiedzi na pytania. Kolejne ko-

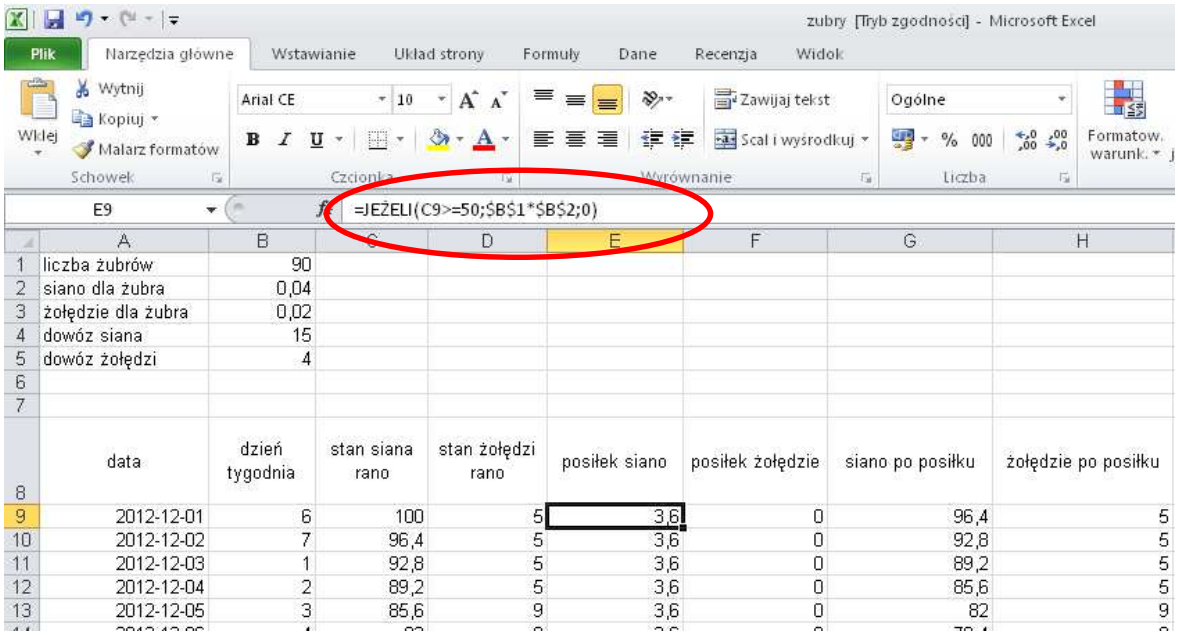
lumny w arkuszu to: poranny stan siana oraz poranny stan żołędzi przed posiłkiem żubrów, ilość siana i żołędzi zużytych podczas posiłku żubrów, stan siana i żołędzi po posiłku żubrów oraz wieczorny stan siana i żołędzi po dostawie.

Aby wyznaczyć dzień tygodnia odpowiadający danej dacie, korzystamy z funkcji `DZIEŃ.TYG`, która zwraca liczbę od 1 do 7 na podstawie danej daty. Należy pamiętać, że drugi parametr pozwala dokładnie sprecyzować, w jaki sposób numerować dni. W naszym przypadku parametr ten jest równy 2, co odpowiada numeracji od poniedziałku = 1 do niedzieli = 7.



	A	B	C	D	E	F
1	liczba żubrów	90				
2	siano dla żubra	0,04				
3	żołędzie dla żubra	0,02				
4	dowóz siana	15				
5	dowóz żołędzi	4				
6						
7						
8	data	dzień tygodnia	stan siana rano	stan żołędzi rano	posiłek siano	posiłek żoł
9	2012-12-01	6				
10	2012-12-02	7				
11	2012-12-03	1				
12	2012-12-04	2				
13	2012-12-05	3				
14	2012-12-06	4				
15	2012-12-07	5				

W pierwszym dniu stan siana rano był równy 100 ton, a stan żołędzi był równy 5 ton i takie liczby wpisujemy w arkuszu w komórkach C9 oraz D9. Jeżeli zapas siana w magazynie wynosi co najmniej 50 ton, to żubry jedzą siano, w przeciwnym razie, dopóki zapas siana nie będzie uzupełniony do co najmniej 50 ton, żubry są karmione tylko żołędziami. Ilość zjedzonego siana odpowiada formule: `=JEŻELI(C9>=50;$B$1*$B$2;0)`, zaś ilość zjedzonych żołędzi odpowiada formule: `=JEŻELI(C9<50;$B$1*$B$3;0)`. Formuły te uwzględniają adresowanie bezwzględne (komórki od B1 do B3), co znacznie ułatwia symulację.

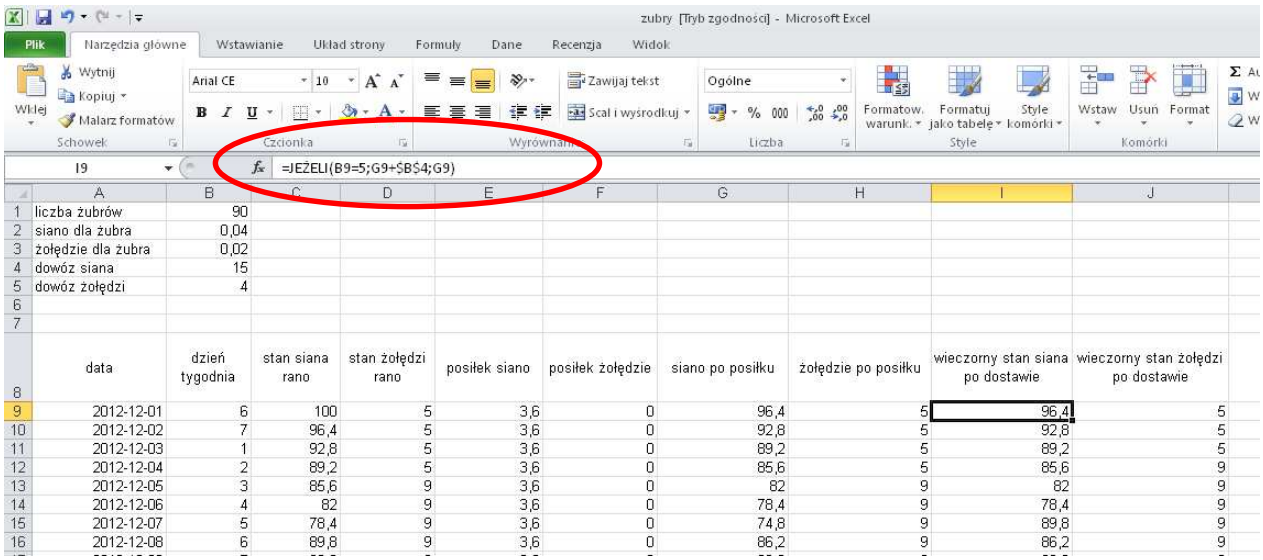


zubry [Tryb zgodności] - Microsoft Excel

E9: =JEŻELI(C9>=50;$B$1*$B$2;0)

	A	B	C	D	E	F	G	H
1	liczba żubrów	90						
2	siano dla żubra	0,04						
3	żołędzie dla żubra	0,02						
4	dowóz siana	15						
5	dowóz żołędzi	4						
6								
7								
8	data	dzień tygodnia	stan siana rano	stan żołędzi rano	posiłek siano	posiłek żołędzie	siano po posiłku	żołędzie po posiłku
9	2012-12-01	6	100	5	3,6	0	96,4	5
10	2012-12-02	7	96,4	5	3,6	0	92,8	5
11	2012-12-03	1	92,8	5	3,6	0	89,2	5
12	2012-12-04	2	89,2	5	3,6	0	85,6	5
13	2012-12-05	3	85,6	9	3,6	0	82	9

W komórce G9 obliczamy siano pozostałe w magazynie po posiłku żubrów: `=C9-E9`, analogicznie stan żołędzi pozostałych po posiłku obliczamy w komórce H9: `=D9-F9`. Dostawy siana odbywają się w każdy piątek po posiłku żubrów, a dostawy żołędzi w każdy wtorek, również po posiłku żubrów. Wieczorny stan zapasów obliczamy:
dla siana: `=JEŻELI(B9=5;G9+$B$4;G9)`, dla żołędzi: `=JEŻELI(B9=2;H9+$B$5;H9)`.



zubry [Tryb zgodności] - Microsoft Excel

I9: =JEŻELI(B9=5;G9+$B$4;G9)

	A	B	C	D	E	F	G	H	I	J
1	liczba żubrów	90								
2	siano dla żubra	0,04								
3	żołędzie dla żubra	0,02								
4	dowóz siana	15								
5	dowóz żołędzi	4								
6										
7										
8	data	dzień tygodnia	stan siana rano	stan żołędzi rano	posiłek siano	posiłek żołędzie	siano po posiłku	żołędzie po posiłku	wieczorny stan siana po dostawie	wieczorny stan żołędzi po dostawie
9	2012-12-01	6	100	5	3,6	0	96,4	5	96,4	5
10	2012-12-02	7	96,4	5	3,6	0	92,8	5	92,8	5
11	2012-12-03	1	92,8	5	3,6	0	89,2	5	89,2	5
12	2012-12-04	2	89,2	5	3,6	0	85,6	5	85,6	9
13	2012-12-05	3	85,6	9	3,6	0	82	9	82	9
14	2012-12-06	4	82	9	3,6	0	78,4	9	78,4	9
15	2012-12-07	5	78,4	9	3,6	0	74,8	9	89,8	9
16	2012-12-08	6	89,8	9	3,6	0	86,2	9	86,2	9

Aby dokończyć symulację pojedynczego dnia, należy wieczorny stan siana i żołędzi przenieść odpowiednio na poranne stany, tzn. uzupełnić komórki C10 i wpisać: `=I9`, a w D10 wpisać: `=J9`. Teraz wystarczy skopiować formuły we wszystkich kolumnach, aż do dnia 28 lutego 2013 r. Tak przygotowany arkusz pozwoli odpowiedzieć na wszystkie pytania w zadaniu.

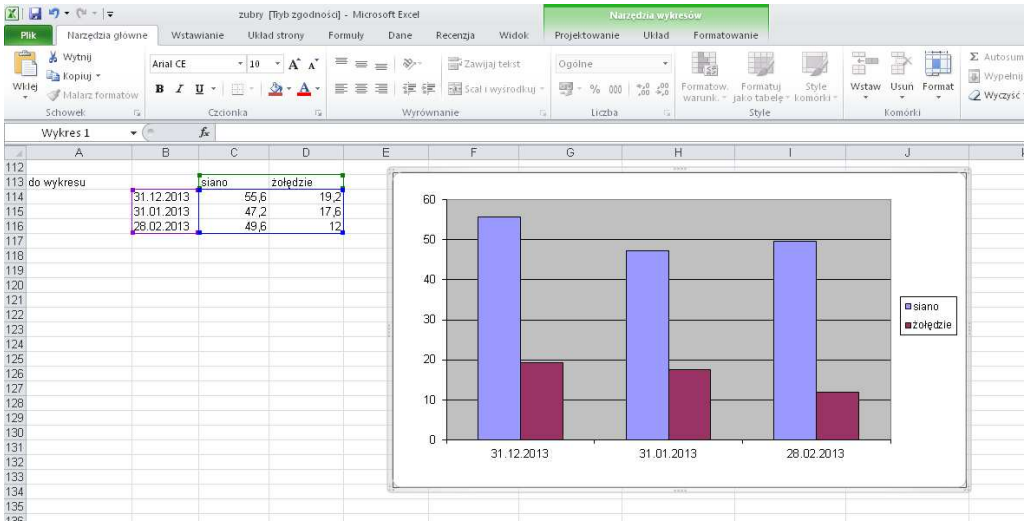
Aby wykonać polecenie a), wystarczy policzyć ile piątków i wtorków znalazło się w rozpatrywanym okresie. Uzyskujemy to dzięki funkcji `LICZ.JEŻELI`, np. dla siana mamy `=LICZ.JEŻELI(B9:B98;5)`.

Podpunkt b) sprawdza, czy poprawnie wykonaliśmy symulację. Rozwiązane wystarczy odczytać z arkusza. Dnia 28 grudnia 2012 r. żubry po raz pierwszy będą karmione żołędziami, po-

nieważ w kolumnie odpowiadającej posiłkowi składającemu się z siana w tym dniu po raz pierwszy pojawia się zero.

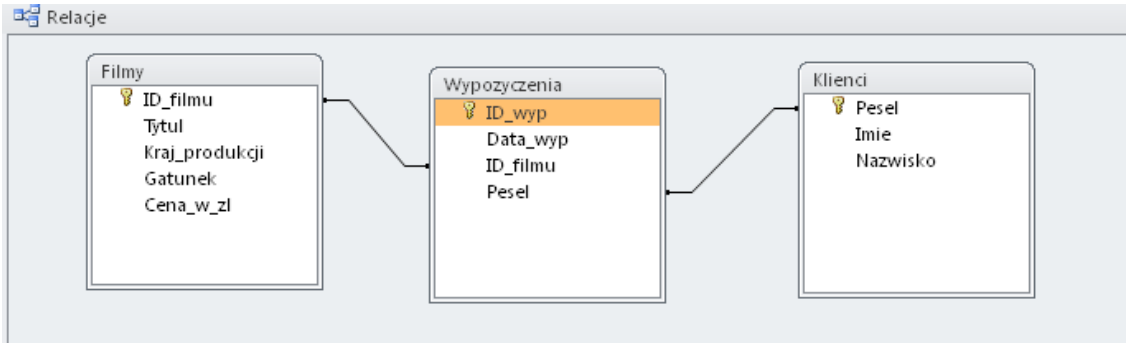
Zadanie c) również kontroluje wykonanie symulacji. Aby sprawdzić, ile razy żubry były karmione sianem, wystarczy skorzystać z funkcji `LICZ.JEŻELI` i policzyć, ile razy w kolumnie F pojawiła się liczba zero (co oznacza, że żubry nie były karmione żołędziami). Analogicznie zadanie wykonujemy dla karmienia żołędziami.

W poleceniu d) należy utworzyć zestawienie porannych stanów zapasów pożywienia dla żubrów oraz sporządzić wykres kolumnowy. Należy pamiętać o prawidłowym zaznaczeniu zakresu danych i opisaniu wykresu w sposób umożliwiający odczytanie stanu pożywienia w wybranym dniu.



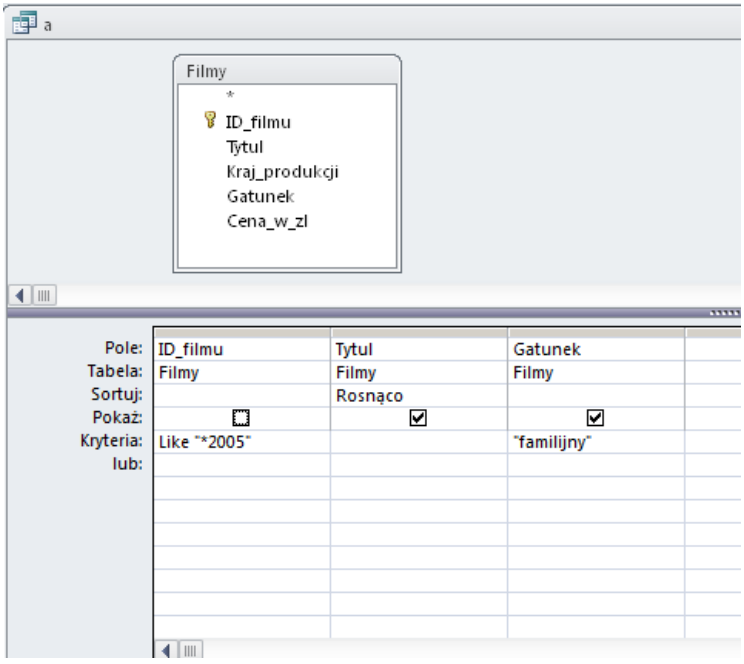
Rozwiązanie ostatniego podpunktu polega na sprawdzeniu, o ile maksymalnie można powiększyć stado żubrów, aby można było je wyżywić według zasad podanych w zadaniu. Dzięki wykorzystaniu adresowania bezwzględnego wystarczy zmieniać wartości w komórce B1. Ostatecznie okaże się, że stado może liczyć maksymalnie 95 osobników, zatem można je powiększyć o 5 żubrów.

Zadanie 5 można rozwiązać zarówno z wykorzystaniem arkusza kalkulacyjnego jak i bazy danych. Do rozwiązania zadania wykorzystamy program MS Access 2010. Na początku należy zaimportować dane z wszystkich plików tekstowych do tabel (Dane zewnętrzne → Importowanie i łączenie → plik tekstowy) . Dzięki korzystaniu z kreatora tabele bazy danych będą mieć takie nazwy, jak pliki tekstowe, a nazwy kolumn – jak wiersze nagłówkowe każdego z plików. Podczas importu należy pamiętać o tym, żeby numer pesel z pliku osoby.txt ustawić w typie danych jako podwójna precyzja lub jako Tekst, a datę wypożyczenia z pliku `wypożyczenia.txt` – jako typ Data/Godzina. Następnie ustalamy relacje między tabelami. Efektem jest następujący schemat bazy danych:

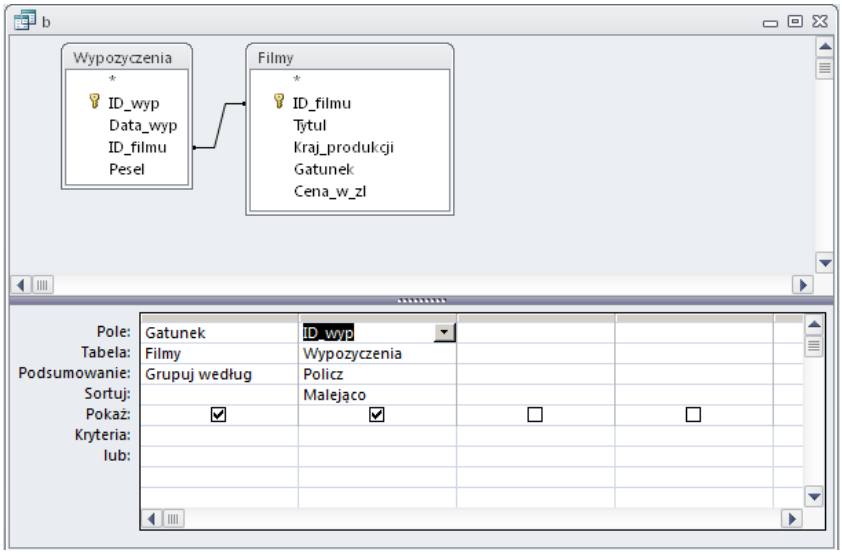


Rozwiązując kolejne podpunkty zadania (a – e) będziemy tworzyć odpowiednie kwerendy.

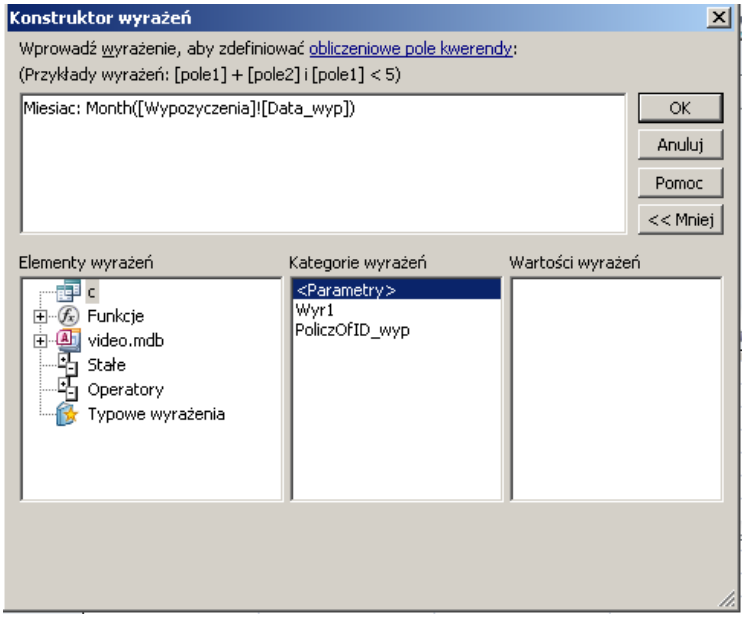
W zadaniu a) wykorzystujemy tylko jedną tabelę `Filmy`, z której należy wybrać filmy familijne wyprodukowane w roku 2005. W kryteriach pola `Gatunek` wpisujemy *`familijny`*, a w kryteriach pola `ID_filmu` wpisujemy „`Like "*2005"` (4 ostatnie cyfry identyfikatora tworzą rok produkcji). Utworzone zestawienie ma być posortowane, więc wybieramy rosnące sortowanie dla pola `Tytul`.



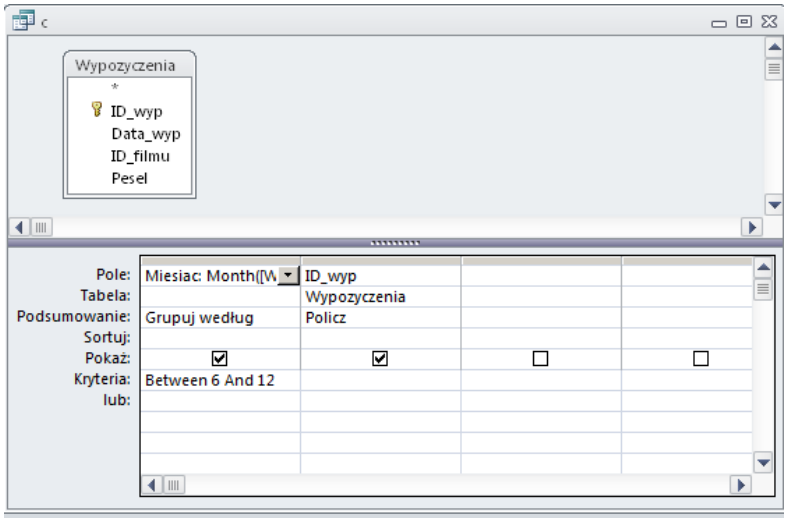
W poleceniu b) należy podać nazwę gatunku filmu o największej popularności wraz z liczbą zamówień wszystkich filmów tego gatunku, dlatego korzystamy z dwóch tabel: `Wypozyczenia` oraz `Filmy`. Z tabeli `Filmy` wybieramy gatunki filmów, które grupujemy. Aby uzyskać liczbę wypożyczeń dla każdego gatunku, musimy policzyć identyfikatory wypożyczeń (funkcja `Policz`). Takie zestawienie sortujemy malejąco – odpowiedź znajduje się wtedy w pierwszym wierszu rozwiązania. Poniżej przedstawione jest okno projektu kwerendy.



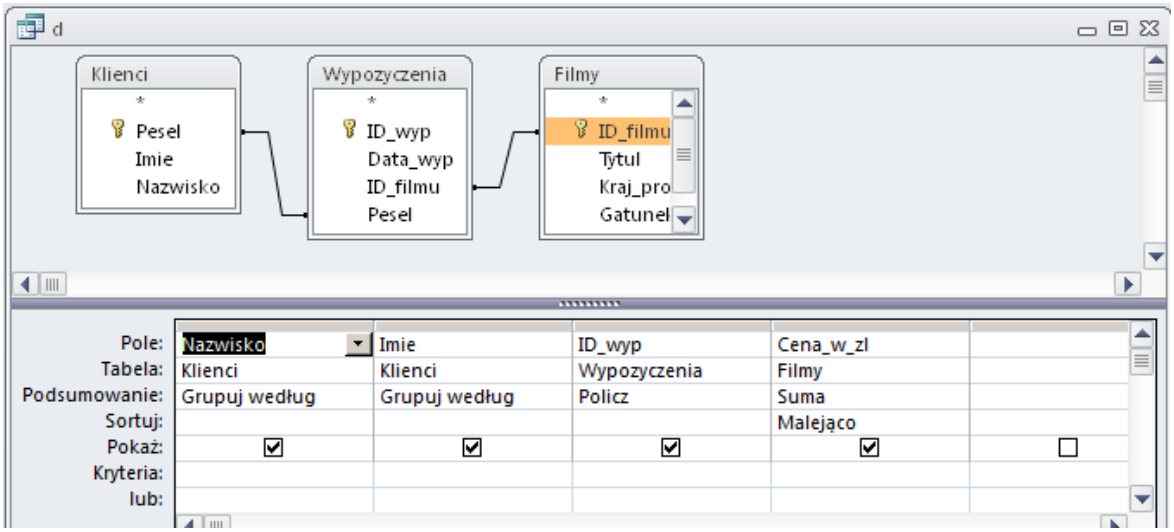
Podpunkt c) wymaga skorzystania z tabeli `Wypozyczenia`, w której znajdują się wszystkie dane potrzebne do rozwiązania zadania. Brakuje w niej jedynie pola, które będzie identyfikowało miesiąc wypożyczenia, dlatego należy dodać pole wyliczane. Takie pola można w prosty sposób tworzyć za pomocą konstruktora wyrażeń (Pole → prawy przycisk myszy → Konstruuj), w którym wybieramy potrzebne elementy. W naszym przypadku będzie to funkcja `Month`, której parametrem jest `Data_wyp` z tabeli `Wypozyczenia`. Na poniższym rysunku przedstawiono widok Konstruktora wyrażeń.



Musimy jeszcze pamiętać, że interesują nas miesiące od czerwca do grudnia, więc w kryteriach wpisujemy np. „`Between 6 And 12`” albo „`>5`”. Aby wyznaczyć liczbę wypożyczeń w każdym miesiącu, grupujemy dane według nowo stworzonego pola i ponownie korzystamy z funkcji Policz, zliczając identyfikatory wypożyczeń w każdej grupie. Szczegółowy projekt kwerendy jest przedstawiony poniżej.



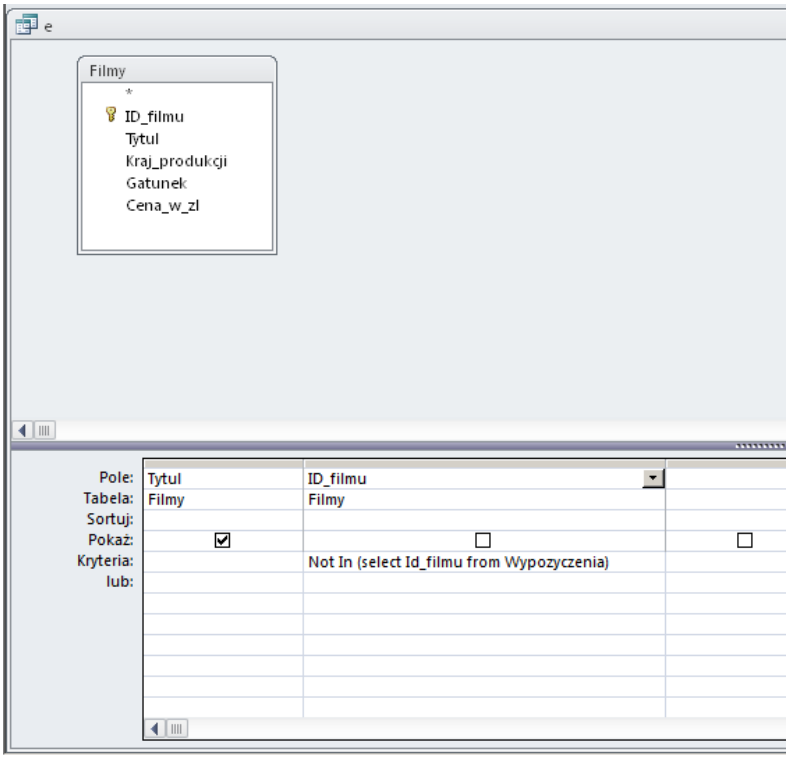
Dopiero polecenie d) zadania wymaga od nas wykorzystania wszystkich trzech tabel bazy danych (imię i nazwisko klienta, który łączeni zapłacił najwięcej za wszystkie wypożyczone filmy) Z tabeli `Klienci` wybieramy nazwiska i imiona osób, z tabeli `Wypozyczenia` wybieramy pole `ID_wyp`, które zliczamy przy pomocy funkcji `Policz`. Ponieważ dodatkowo interesuje nas, jaką kwotę zapłacił klient, z tabeli Filmy wybieramy pole Cena_w_zl i korzystamy z funkcji Suma. Następnie stosujemy sortowanie malejące względem cen, dzięki czemu odpowiedź do zadania znajdzie się w pierwszym wierszu otrzymanego zestawienia.



W poleceniu e) z tabeli `Filmy` musimy wybrać te tytuły, które nie zostały wypożyczone, czyli takie, których identyfikatory nie znajdują się w tabeli Wypozyczenia. Takie zapytanie wystarczy zapisać w języku SQL:

```
SELECT Filmy.Tytul FROM Filmy
WHERE (((Filmy.Id_Filmu) Not In (select ID_Filmu From Wypozyczenia)));
```

Jeżeli chcemy jednak nadal korzystać z widoku projektu, w kryteriach umieszczamy zapis: `Not In (select ID_Filmu From Wypozyczenia)` (zob. rysunek poniżej).



Zadanie 6 jest zadaniem programistycznym. Polecenia a) oraz c) często pojawiały się w zadaniach maturalnych w tej samej lub podobnej postaci. Polecenie b) oraz polecenie c) sprawdzają umiejętność algorytmicznego myślenia i wykorzystania języka programowania do zapisu algorytmu.

W rozwiązaniu zadania wykorzystamy język C++. Ponieważ wiemy, ile elementów znajduje się w pliku, do pobierania kolejnych elementów użyjemy pętli FOR. W rozwiązaniu wykorzystamy także klasę fstream do obsługi plików oraz klasę string do obsługi napisów. Należy podkreślić, że każde z poleceń może mieć wiele metod rozwiązania, a my przedstawimy tylko jedno przykładowe.

W poleceniu a) trzeba policzyć, ile jest takich elementów w pliku, że pierwsza cyfra równa się ostatniej cyfrze. Jest to polecenie bardzo łatwe. Najprościej jest pobierać z pliku kolejne elementy jako napisy i sprawdzać, czy pierwszy znak jest równy ostatniemu. Poniżej prezentujemy fragment kodu programu:

```
int i;
ifstream plik;
plik.open("dane.txt");
string wiersz;
int ile=0;
for(i=1; i<=5000; i++)
{
    plik>>wiersz;
    if (wiersz[0]==wiersz[wiersz.length()-1])
      ile++;
  }
plik.close();
```

Polecenie b) różni się od punktu a) jedynie tym, że porównywanie cyfr dotyczy reprezentacji dziesiętnej liczb z pliku, zatem po pobraniu liczby zamieniamy ją na system dziesiętny wykorzy-

stując np. schemat Hornera. Musimy jeszcze pamiętać o tym, jak zamienić w C++ cyfrę zapisaną jako znak na cyfrę zapisaną jako liczbę. Najprościej można to zrobić odejmując od znaku napisu liczbę 48 (`x[j]-48`), ponieważ kod ASCII cyfry zero ma wartość równą 48. Poniżej prezentujemy prostą funkcję zamiany liczby z systemu ósemkowego na dziesiętny:

```
int oct2dec(string x)
{
   int dec=0; // liczba dziesietna
   int j;
   for (j=0; j<=x.length()-1; j++)
       dec=8*dec+(x[j]-48); //zamiana znaku na liczbę
   return dec;
}
```

Teraz wystarczy zamienić liczbę dziesiętną na napis i wykorzystać kod programu z punktu a). My dla odmiany przedstawimy inny sposób. Aby uzyskać ostatnią cyfrę liczby zapisanej dziesiętnie, wystarczy obliczyć jej resztę z dzielenia przez 10. Żeby mieć dostęp do pierwszej cyfry liczby, trzeba pozbywać się jej kolejnych cyfr, co daje operacja dzielenia całkowitego przez 10, którą wykonujemy tak długo, jak długo liczba jest większa od 9. Na końcu porównujemy pierwszą i ostatnią cyfrę:

```
for(i=1; i<=5000; i++)
 {
    plik>>wiersz;
    liczba=oct2dec(wiersz);// wlasna funkcja

    last=liczba%10; //ostatnia cyfra
    while (liczba>9)
       liczba=liczba/10;
    if (last==liczba) //liczba jest juz rowna pierwszej cyfrze
       ile++;
 }
```

Podpunkt c) sprawdza znajomość algorytmu znajdowania minimalnego i maksymalnego elementu w zbiorze oraz umiejętność sprawdzenia, czy liczba jest niemalejąca (tzn. czy każda kolejna cyfra liczby nie jest mniejsza niż poprzednia jej cyfra). Każdą liczbę z pliku ponownie potraktujemy jako napis. Sprawdzamy kolejne elementy napisu i jeżeli kolejny znak w napisie okaże się mniejszy niż znak poprzedni, funkcja zwróci wartość false. Kolejne znaki napisu można sprawdzać w pętli for, co na pewno jest ułatwieniem dla zdających. W naszym rozwiązaniu celowo zastosowaliśmy pętlę while i zmienną logiczną odp, która jest odpowiedzialna za jej przerwanie. Może się przecież okazać, że już na samym początku liczba, którą sprawdzamy, nie będzie spełniała postawionego w zadaniu warunku, zatem nie ma sensu, aby pętla nadal się wykonywała. Poniżej przedstawiamy funkcję sprawdzającą, czy liczba jest niemalejąca:

```
bool punkt_c(string wiersz)
{
bool odp=true;
int i=0;
while (odp && i<wiersz.length()-1)
{
  if (wiersz[i] >wiersz[i+1])
     odp=false;
  i++;
 }
```

Teraz wystarczy użyć powyższej funkcji dla każdej liczby z pliku – jeżeli wynikiem funkcji będzie true, to sprawdzimy, czy liczba ta może być liczbą minimalną lub maksymalną. Tutaj można wykorzystać klasyczny algorytm znajdowania minimalnego (lub maksymalnego) elementu w zbiorze. Na przykładzie minimum: na początku ustalamy, że pierwszy element w zbiorze jest najmniejszy i jeśli jakikolwiek inny element będzie mniejszy, to będzie on nowym kandydatem na minimum. Jeśli będziemy traktować liczby jak napisy, może to być zadanie trochę trudniejsze, ponieważ podczas porównywania napisów brany jest porządek leksykograficzny. Najpierw trzeba porównywać długości napisów. Jeżeli kolejny napis będzie miał długość mniejszą niż zapamiętane minimum, to będzie on kandydatem na minimum i dodatkowo trzeba zmienić minimalną długość. Jeżeli kolejny napis będzie miał taką samą długość jak zapamiętane minimum, to wystarczy porównać ten napis z minimum i ewentualnie zmienić minimum. Analogicznie postępujemy dla maksimum.

Musimy zwrócić uwagę na jeszcze jedną rzecz: w klasycznym algorytmie najczęściej ustalamy pierwszy element zbioru jako minimalny (lub maksymalny). W naszym przypadku jest to dodatkowo trudność, ponieważ nie wiadomo, który element jako pierwszy będzie liczbą niemalejącą – do ustalenia takiej sytuacji potrzebna byłaby dodatkowa zmienna. Jednak my możemy skorzystać z warunków przedstawionych w zadaniu: liczby w pliku są z przedziału 10_8 do 2000000_8, dlatego na początku można przyjąć, że minimum jest równe 2000000_8 (każda inna liczba nie będzie większa od wskazanej), a maksimum jest równe 10_8 (każda inna liczba nie będzie mniejsza od wskazanej). Poniżej prezentujemy fragment kodu źródłowego:

```
string min="2000000", max="10"; //liczby traktujemy jak napisy
int dl_min=7; dl_max=2;
int ile=0;
for(i=1; i<=5000; i++)
{
   plik>>wiersz;
   if (c(wiersz)) //liczba niemalejaca
   {
    ile++;
    if (wiersz.length()<dl_min)
    {
        min=wiersz;//nowe min
        dl_min=wiersz.length(); //zmiana min. dlugosci
     }
     if (wiersz.length()==dl_min)    //ta sama dlugosc
        if (wiersz<min) min=wiersz; //tylko nowe min.
     if (wiersz.length()>dl_max)
     {
         max=wiersz; //nowe max
         dl_max=wiersz.length(); //zmiana max.dlugosci
     }
     if (wiersz.length()==dl_max)    //ta sama dlugosc
        if (wiersz>max) max=wiersz; //tylko nowe max.
     }
}
```