MATURA 2021 – czerwiec

https://www.korepetycjezinformatyki.pl/rozwiazania-matura-z-informatyki-maj-2021/

Zadanie 1 – Cyfrowe dopełnienie

Zadanie 1. Cyfrowe dopełnienie

Niech *n* będzie nieujemną liczbą całkowitą, której najbardziej znacząca cyfra w zapisie dziesiętnym jest większa od 0 i mniejsza od 9. Cyfrowym dopełnieniem liczby *n* nazywamy liczbę całkowitą *d*, której zapis dziesiętny otrzymujemy z zapisu dziesiętnego liczby *n* przez zamianę każdej cyfry tego zapisu na cyfrę, która jest jej uzupełnieniem do 9.

Przykład:
Cyfrowym dopełnieniem liczby 2021 jest liczba 7978.

Zadanie 1 Cyfrowe dopełnienie z arkusza maturalnego z informatyki maj 2021

Zacznijmy od przykładów liczb (nazwanych n) i ich cyfrowych dopełnień (nazwanych d). Zgodnie z założeniem najbardziej znacząca cyfra (a więc pierwsza od lewej) ma zawierać się w przedziale od 1 do 8. Jest to pierwszy i główny wymóg. Należy tu pamiętać, że cyfry 0 i 9 mogą pojawiać się w dalszej części liczby.

Prawidłowe liczby n będą więc wyglądać w ten sposób: 1, 8000, 545454 czy 89999.

Cyfrowe dopełnienie stanowi taka liczba d dla której suma kolejno każdej cyfry da 9, najłatwiej zobrazować to na przykładzie:

Liczba n:	12345	100	545454
Dopełnienie:	87654	899	454545
Suma:	99999	999	999999

Rozumiejąc sposób tworzenia liczb n i d, możemy przystąpić do właściwych zadań.

Zadanie 1.1.

Zadanie 1.1. (0–2)

Podaj czterocyfrową liczbę *n* taką, że wartość bezwzględna różnicy liczby *n* i jej cyfrowego dopełnienia *d* jest:

a) najmniejsza $n =$ _____

b) największa $n =$ _____

Zadanie 1.1 Cyfrowe dopełnienie z arkusza maturalnego z informatyki maj 2021

Naszym zadaniem jest znalezienie przypadków granicznych dla czterocyfrowej liczby n: maksimum i minimum wartości bezwzględnej różnicy liczby n i dopełnienia d.

Brzmi to dość niepokojąco, jednak rozbijmy to na części pierwsze, a wszystko stanie się jasne.
Wartość bezwzględna (oznaczana w zapisie przez abs, od ang. absolute) sprawia, że wynik zawsze będzie nieujemny, a więc różnica zapisana wewnątrz staje się przemienna.

Dla przykładu: abs(900-100) daje wynik 800, a abs(100-900) -> abs(-800) -> 800. Wyniki są takie same.

Kiedy wartość bezwzględna z różnicy liczb n i d będzie najmniejsza? Wtedy, gdy będą one leżeć najbliżej siebie. Wykorzystując wiedzę dotyczącą tworzenia liczb n i d musimy teraz znaleźć te liczby.

Dla n = 5000 nasze d jest równe 4999, bo 5000+4999 = 9999, tak więc różnica jest równa 1.
Możemy spróbować podstawić pod n liczbę 4999, w tym wypadku okaże się, że to d jest równe 5000.
W ten sposób znaleźliśmy dwie liczby, które są prawidłowymi odpowiedziami, ponieważ abs(5000-4999) == abs(4999-5000) i niemożliwe jest uzyskanie mniejszej różnicy.

Zostaje do obliczenia wartość maksymalna. Przy wyznaczaniu maksimum i minimum funkcji zawsze warto sprawdzić, co się stanie, gdy pod liczbę podstawimy wartości graniczne. W naszym wypadku z założeń wynika, że stanowią je liczby 1000 oraz 8999 (pod pierwszą cyfrę nie możemy podstawić 0 ani 9).

Dla n = 1000 wychodzi, że d będzie równe 8999, abs(1000-8999) = 7999.
Wynik będzie taki sam, jeśli pod n podstawimy 8999.

W ten sposób uzyskaliśmy odpowiedzi: 5000 lub 4999 oraz 1000 lub 8999.

Zadanie 1.2.

Zadanie 1.2. (0–4)

W postaci pseudokodu lub w wybranym języku programowania napisz algorytm, który dla dodatniej liczby całkowitej *n* obliczy jej cyfrowe dopełnienie *d*. O liczbie *n* wiadomo, że jej najbardziej znacząca cyfra jest większa od 0 i mniejsza od 9.

Uwaga: Twój algorytm może używać wyłącznie zmiennych przechowujących liczby całkowite oraz może operować wyłącznie na liczbach całkowitych. W zapisie algorytmu możesz korzystać tylko z instrukcji sterujących, operatorów arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, dzielenia całkowitego i reszty z dzielenia; operatorów logicznych, porównań i instrukcji przypisywania lub samodzielnie napisanych funkcji i procedur wykorzystujących powyższe operacje. **Zabronione** jest używanie funkcji wbudowanych dostępnych w językach programowania. Nie wolno w szczególności korzystać z żadnych funkcji zamiany z typu znakowego lub napisowego na liczbowy i odwrotnie.

Specyfikacja:

Dane:

n – dodatnia liczba całkowita taka, że jej najbardziej znacząca cyfra jest większa od 0 i mniejsza od 9

Wynik:

d – dodatnia liczba całkowita, cyfrowe dopełnienie liczby *n*

Zadanie 1.2 Cyfrowe dopełnienie z arkusza maturalnego z informatyki maj 2021

Sposobów na rozwiązanie tego typu zadań jest wiele. Omówimy tu dwa rozwiązania, które są stosunkowo proste do zrozumienia i warto o nich wiedzieć, aby dobrze przygotować się na przyszłość.

Pierwszy sposób.

Aby napisać algorytm do tego problemu, należy znać właściwości operatorów dzielenia całkowitego (oznaczonego jako **div**) oraz reszty z dzielenia (**mod**).

Dzielenie całkowite polega na odrzuceniu reszty przy dzieleniu. Dla przykładu: 999 div 10 daje wynik **99**.
Reszta z dzielenia wykonuje dokładnie to, co ma w nazwie: wyznacza resztę z dzielenia. Dla działania 999 mod 10 otrzymujemy wynik **9** (999/10 to 99 z resztą równą **9**).

Wykorzystując te dwa operatory, ustalmy kroki, jakie należy podjąć, aby obliczyć liczbę d.

Najlepszą opcją jest wzięcie kolejno każdej cyfry i odjęcie jej od 9: w ten sposób najpewniej sami liczyliście liczbę d wcześniej. Operujemy jednak na wartościach liczbowych, dlatego kolejność działań musi następować od końca:

- wyznaczamy ostatnią cyfrę wykorzystując resztę z dzielenia przez 10 (n mod 10)
- odejmujemy tę cyfrę od 9 i wynik odejmowania przemnażamy, aby uzyskać wartość dziesiętną, setną, tysięczną itd.
- dodajemy uzyskaną liczbę do zmiennej oznaczającą liczbę d

Czynności te powinniśmy powtórzyć dla każdej cyfry, dlatego po wykonaniu tych działań podzielimy całkowicie liczbę n przez 10. Pozwoli to nam na przejście do kolejnej cyfry zaczynając od końca.

Aby lepiej zobrazować przebieg algorytmu, możemy stworzyć tabelkę dla np. n = 4561:

Lp.	**Wartość n**	**Wartość d**	**n mod 10**	**9-(n mod 10)**	**i**	**n div 10**
1	4561	0	1	8	1	456
2	456	8	6	3	10	45
3	45	38	5	4	100	4
4	4	438	4	5	1000	0
5	0	5438	–	–	–	–

Znamy sposób na uzyskanie liczby d. Przejdźmy więc do pseudokodu:

```
d ← 0
i ← 1 // zaczynamy od prawej strony, a więc od jedności
dopóki n ≠ 0 wykonuj
    reszta ← n mod 10
    wynik ← (9 – reszta) * i
    d ← d + wynik
    n ← n div 10
    i ← i * 10 // po pierwszy wykonaniu z jedności przechodzimy na dziesiątki itd.
zwróć d i zakończ
```

Drugi sposób.

W tym rozwiązaniu wykorzystamy sposób wyznaczania d polegający na odejmowaniu liczby n od liczby składającej się z samych dziewiątek. Dla przykładu:
n = 3579, to d = 9999 – 3579 = 6420
n = 11, to d = 99 – 11 = 88

Musimy tu zadbać o wyznaczenie zmiennej z odpowiednią liczbą dziewiątek. Aby to zrobić, ponownie skorzystamy z dzielenia całkowitego, tak więc znajomość jego działania wydaje się nieunikniona:

```
dziewiatki ← 0
i ← 1
oryginalne_n = n // niżej zmieniać będziemy wartość n, więc zapiszmy sobie jej początkową wartość
dopóki n ≠ 0 wykonuj
    dziewiatki ← dziewiatki + 9 * i
    n ← n div 10
    i ← i * 10
d ← dziewiatki – oryginalne_n
zwróć d i zakończ
```

Zadanie 2 – Analiza algorytmu

Zadanie 2. Analiza algorytmu

Niech *n* będzie nieujemną liczbą całkowitą, a *T*[1..*n*] – tablicą zawierającą *n* liczb całkowitych. Dla *n* = 0 tablica *T* jest pusta (nie zawiera żadnego elementu).
Wykonaj analizę poniżej zapisanej funkcji *d*(*x*), która rozszerza tablicę *T* o liczbę całkowitą *x*, a następnie przeprowadza pewną reorganizację zawartości tej tablicy.

```
d(x):
    n ← n + 1
    T[n] ← x
    s ← n
    dopóki ((s div 2) ≥ 1) oraz (T[s] > T[s div 2]) wykonuj
        pom ← T[s]
        T[s] ← T[s div 2]
        T[s div 2] ← pom
        s ← s div 2
```

Uwaga: w tym zadaniu przyjmujemy, że:

- tablica *T* może być powiększana;
- jeśli wartość lewego argumentu operatora **oraz** jest równa *fałsz*, to wartość prawego argumentu nie jest wyliczana;
- *div* jest operatorem oznaczającym część całkowitą z dzielenia.

Zadanie 2 Analiza algorytmu z arkusza maturalnego z informatyki maj 2021

W zadaniach z analizy algorytmu sposobem, który wydaje się być najbardziej odporny na pomyłki, jest tworzenie tabelki, dzięki której śledzimy etapy wykonywania algorytmu linijka po linijce.

Każda kolumna oznacza wartość zmiennej bądź sprawdzany warunek, a każdy wiersz jest kolejnym wykonaniem pętli. Pozwala to na bezpieczne prześledzenie działania algorytmu.

Warto jednak najpierw dokładnie przeczytać kroki zapisane w pseudokodzie i znaleźć charakterystyczne operacje.

Przed pętlą w funkcji zachodzą trzy operacje:

- zwiększenie wartości n o 1 (n ← n + 1)
- dodanie zmiennej x do tablicy T (T[n] ← x)
- przypisanie zmiennej s aktualnej wartości n (która wcześniej została powiększona o 1) (s ← n)

Widzimy tu mechanizm dodawania nowej wartości na koniec tablicy. Do zmiennej s zapisywany jest indeks tej nowej wartości.

Kolejną część algorytmu stanowi pętla **dopóki**. Przed wykonaniem jej sprawdzane są dwa warunki:

- czy wynik z dzielenia całkowitego s przez 2 jest większy lub równy 1
- czy wartość w tablicy T pod indeksem s jest większa od wartości pod indeksem s div 2

Oba warunki muszą zostać spełnione, aby instrukcje wewnątrz pętli zostały wykonane, ponieważ łączy je spójnik **oraz**.

Wewnątrz pętli zachodzi popularna w informatyce operacja: zamiana dwóch wartości w tabeli z wykorzystaniem zmiennej pomocniczej. Dokonuje się w tym miejscu przestawienie wartości w tabeli T pod indeksem s z wartością pod indeksem s div 2.
Po zamianie dzielimy całkowicie zmienną s przez 2, a więc zmniejszamy jej wartość o połowę i odrzucamy ewentualną resztę z dzielenia (np. dla s = 5, po wykonaniu s ← s div 2 zmienna ta przyjmie wartość 2). W tym momencie możemy zauważyć, że zmienna s przechowuje przez cały czas aktualny indeks zmiennej, którą dodaliśmy do tablicy na początku (zmienna x).

Algorytm ten polega więc na ciągłym przestawianiu w tablicy T wartości zmiennej x, jeśli po środku stoi wartość mniejsza od niej.

Zadanie 2.1.

Zadanie 2.1. (0–2)
Uzupełnij tabelę – wpisz zawartość tablicy *T* po wykonaniu *d*(*x*) z podanym parametrem *x*:

n	*T*[1..*n*]	*x*	*T* po wykonaniu *d*(*x*)
4	26, 3, 5, –4	5	26, 5, 5, –4, 3
4	36, 15, 17, 3	–5	
7	27, 6, 13, 4, –3, –2, –3	30	

Zadanie 2.1 Analiza algorytmu z arkusza maturalnego z informatyki maj 2021

Zacznijmy od pierwszego, rozwiązanego już przykładu, aby sprawdzić, czy nasz sposób jest prawidłowy.

Dla danych n = 4, T = [26, 3, 5, -4] oraz x = 5 po wykonaniu trzech początkowych operacji, nasze dane wyglądają tak:
n = 5, T = [26, 3, 5, -4, 5], s = 5
Teraz prześledźmy wykonanie pętli:

Lp.	s	s div 2	(s div 2)>=1?	T przed zamianą	T[s]>T[s div 2]	T po zamianie
1	5	2	PRAWDA	[26, 3, 5, -4, 5]	5 > 3 PRAWDA	[26, 5, 5, -4, 3],
2	2	1	PRAWDA	[26, 5, 5, -4, 3]	5 > 26 FAŁSZ	–

Podczas drugiego sprawdzenia nie zachodzi zamiana, ponieważ jeden z warunków (T[s]>T[s div 2]) zwrócił fałsz. Wynik T = [26, 5, 5, -4, 3] zgadza się z podaną w zadaniu odpowiedzią, więc możemy przejść dalej.

Dla drugiego przykładu:
n = 4, T = [36, 15, 17, 3], x = -5
Po wykonaniu początkowych operacji:
n = 5, T = [36, 15, 17, 3, -5], s = 5

Lp.	s	s div 2	(s div 2)>=1?	T przed zamianą	T[s]>T[s div 2]	T po zamianie
1	5	2	PRAWDA	[36, 15, 17, 3, -5]	-5 > 15 FAŁSZ	–

Jak widać, w tym przypadku pętla nie wykona się ani razu, więc odpowiedzią będzie
T = [36, 15, 17, 3, -5]

Dla trzeciego przykładu:
n = 7, T = [27, 6, 13, 4, -3, -2, -3], x = 30
Po wykonaniu początkowych operacji:
n = 8, T = [27, 6, 13, 4, -3, -2, -3, 30], s = 8

Lp.	s	s div 2	(s div 2)>=1?	T przed zamianą	T[s]>T[s div 2]	T po zamianie
1	8	4	PRAWDA	[27, 6, 13, 4, -3, -2, -3, 30]	30 > 4 PRAWDA	[27, 6, 13, 30, -3, -2, -3, 4]
2	4	2	PRAWDA	[27, 6, 13, 30, -3, -2, -3, 4]	30 > 6 PRAWDA	[27, 30, 13, 6, -3, -2, -3, 4]
3	2	1	PRAWDA	[27, 30, 13, 6, -3, -2, -3, 4]	30 > 27 PRAWDA	[30, 27, 13, 6, -3, -2, -3, 4]
4	1	0	FAŁSZ			

Otrzymujemy wynik T = [30, 27, 13, 6, -3, -2, -3, 4]

Z tego przykładu możemy wyciągnąć pewny wniosek. Jeśli wartość zmiennej x przekazywanej do funkcji jest większa od innych elementów w tablicy T, to będzie ona zamieniać się miejscami z innymi liczbami, aż znajdzie się na początku tablicy. Tak właśnie się stało dla x = 30.

Zadanie 2.2.

Zadanie 2.2. (0–2)

Podaj zawartość tablicy *T* po wykonaniu wszystkich sześciu wywołań funkcji *d* kolejno z parametrami: 6, –4, 12, 27, 26, 8, przy początkowo pustej tablicy *T*.

…….

Zadanie 2.2 Analiza algorytmu z arkusza maturalnego z informatyki maj 2021

W tym zadaniu wywołujemy po kolei: d(6), d(-4), d(12)… Jako że początkowo tablica T jest pusta, po pierwszym wywołaniu funkcji: d(6) T będzie równe [6].

Następnie dokładamy kolejny element: -4.
Do rozwiązania tego zadania możemy wykorzystać metodę tabelkową, przedstawię tu

natomiast skrócony opis zachodzących operacji.
Początkowo tablica przybiera postać T = [6, -4]. Zmienna s oznacza indeks 2, bo pod nim znajduje się liczba -4. Sprawdzamy więc, czy -4 > 6. Jest to fałsz, dlatego możemy przejść do kolejnego wykonania funkcji.

Dla x = 12 tablica T = [6, -4, 12]. Tym razem porównujemy 12 z 6, ponieważ 3 div 2 = 1. Liczba 12 jest większa od 6, dlatego zamieniamy je miejscami: T = [12, -4, 6]. W tym momencie s = 1 i nie mamy więcej elementów do sprawdzania, ponieważ zmienna x (czyli 12) trafiła już na początek

Następnie wykonujemy funkcję d(27): T = [12, -4, 6, 27]. 27 jest większe od -4, więc zamieniamy je miejscami: T = [12, 27, 6, -4]. 27 jest też większe od 12, dlatego ląduje ono na początku: T = [27, 12, 6, -4]

Dla d(26) T = [27, 12, 6, -4, 26]. 26 jest większe od 12, czyli T = [27, 26, 6, -4, 12]. 26 nie jest natomiast większe od 27. Tabelę zostawiamy tej formie.

Dla x = 8 tablica T = [27, 26, 6, -4, 12, 8]. 8 > 6, czyli T = [27, 26, 8, -4, 12, 6]. Po lewej stronie od 8 pozostały same większe liczby, co oznacza, że właśnie tak będzie wyglądać ostateczna tabela.

Zadanie 2.3.

Zadanie 2.3. (0–2)

Do początkowo pustej tablicy *T* wstawiono za pomocą funkcji *d* kolejno liczby całkowite od 1 do *k* – 1. Wstawiamy teraz do tablicy *T* kolejną liczbę *k* za pomocą *d*(*k*).
Zapisz, ile razy w trakcie wykonywania *d*(*k*) sprawdzany jest warunek pętli *dopóki*: „((*s* div 2) ≥ 1) oraz (*T*[*s*] > *T*[*s* div 2])" dla podanych wartości *k*.

k	Ile razy sprawdzany jest warunek pętli *dopóki* podczas wykonywania *d*(*k*)?
4	3 razy
16	
1025	

Zadanie 2.3 Analiza algorytmu z arkusza maturalnego z informatyki maj 2021

Aby prawidłowo rozwiązać to zadanie, należy dobrze zrozumieć działanie algorytmu. Z treści zadania wynika, że początkowo w tabeli znajdują się liczby od 1 do k-1. Znaczy to, że dla k = 4 T = [1, 2, 3]. Do tej tablicy dokładamy liczbę k, w tym przypadku będzie to 4. Oznacza to, że przesunięcia będziemy dokonywać do czasu, aż liczba k nie znajdzie się na początki tablicy T.
Tablica ta będzie przyjmować wartości kolejno:

1. [1, 2, 3, 4]: po tym następuje pierwsze sprawdzenie warunki pętli
2. [1, 4, 3, 2]: po zamianie kolejne
3. [4, 1, 3, 2]: i trzecie, ostatnie, ma miejsce, gdy liczba k jest już na początku tablicy

Dla k = 8 sprawdzenia będą miały miejsce gdy k jest na 8., 4., 2. i 1. pozycji.
Podążając tym śladem, możemy wyznaczyć odpowiedzi do zadania:

Dla k = 16 sprawdzenia nastąpią przy indeksie 16, 8, 4, 2 oraz 1. Otrzymujemy 5 sprawdzeń.

Dla k = 1025 warunek pętli zostanie sprawdzony na indeksach 1025, 512 (bo 1025 dzielimy całkowicie na pół), 256, 128, 64, 32, 16, 8, 4, 2 i 1, co daje nam odpowiedź 11.

Zadanie 3 – Test

Zadanie 3.1.

Zadanie 3.1. (0–1)
Dana jest następująca funkcja:

```
funkcja f(n):
    jeżeli n > 0
        wypisz n
        f(n – 2)
        wypisz n
```

1.	W wyniku wywołania *f*(5) otrzymamy ciąg 5 5 5 5 5 5.	**P**	**F**
2.	W wyniku wywołania *f*(6) otrzymamy ciąg 6 4 2 2 4 6.	**P**	**F**
3.	W wyniku wywołania *f*(7) otrzymamy ciąg 7 5 3 1 1 3 5 7.	**P**	**F**
4.	W wyniku wywołania *f*(8) otrzymamy ciąg 8 6 4 2 0 0 2 4 6 8.	**P**	**F**

Zadanie 3.1 Test z arkusza maturalnego z informatyki maj 2021

W tym zadaniu pojawia się rekurencja, czyli wywoływanie funkcji wewnątrz jej samej. Dobrym sposobem na zwizualizowanie rekurencji jest rysowanie drzewa wywołań:

- przed wywołaniem funkcji następuje wypisanie liczby n, co zapiszemy na górze
- następnie wykonuje się kolejna funkcja z argumentem n pomniejszonym o dwa, zapisujemy to po środku
- gdy argument n przestanie być większy od 0, następuje drugie wypisywanie liczby n. Tym razem ma ono jednak miejsce od końca, czyli najmniejszej liczby do największej.

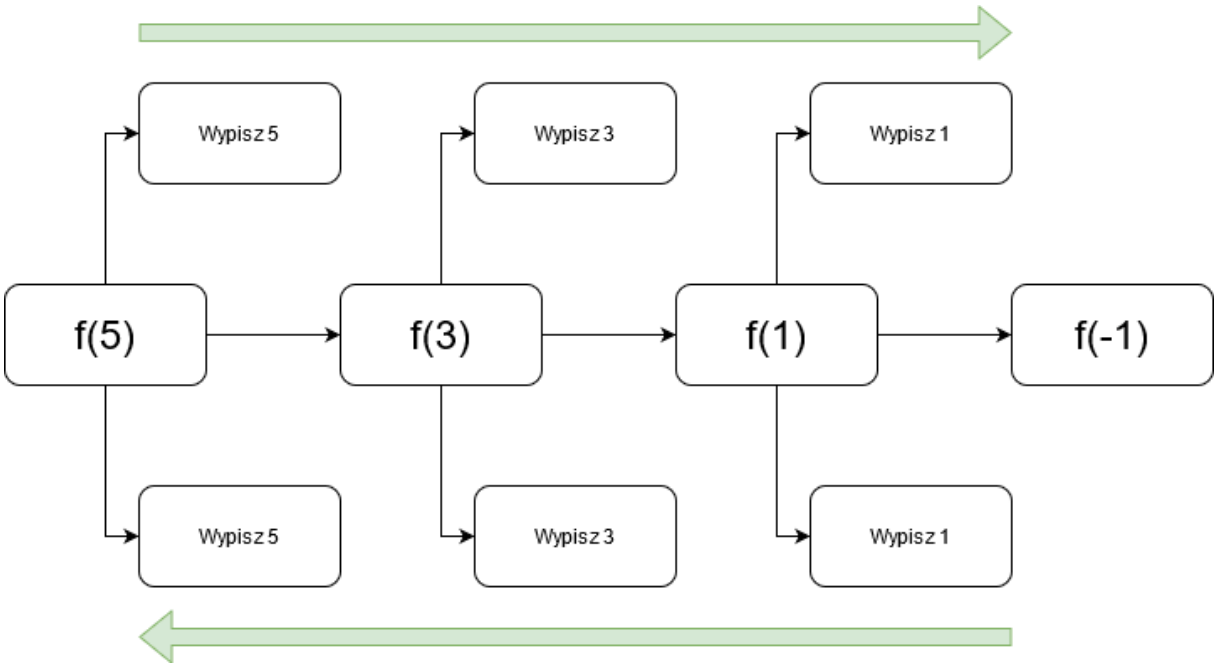


Diagram przedstawiający rekurencję.

Analizując wynik wywołania funkcji f podążamy za strzałkami: zaczynamy od lewej strony z góry (5, 3, 1), a później od prawej strony z dołu (1, 3, 5).
Dla f(-1) nic nie zostaje wypisane, ponieważ -1 nie spełnia warunku n > 0.
Prawidłową odpowiedź stanowi 5 3 1 1 3 5, a więc pierwszy podpunkt jest fałszywy.

Z drugim podpunktem postępujemy podobnie. Znamy już mechanizm działania tej funkcji: najpierw wypisane zostają liczby od n do 1 z odstępami co 2, a później podążamy w drugą stronę.
Zaczynamy od 6, później 4, 2. 0 nie jest wypisywane, ponieważ nie spełnia warunku n > 0.
Następnie te same liczby zostają wypisane od końca.
Odpowiedź 6 4 2 2 4 6 jest prawidłowa, ponieważ spełnia ona wszystkie przedstawione warunki.

Sprawdzamy trzeci podpunkt. W pierwszym przejściu wypisane zostają liczby 7, 5, 3, i 1.
Wypisując teraz liczby w drugą stronę dowiadujemy się, że odpowiedź 7 5 3 1 1 3 5 7 jest także prawdziwa.

W ostatnim przypadku możemy od razu dostrzec, że jest on fałszem. Wszystko za sprawą pojawiających się zer, które są niezgodne z warunkiem n > 0.

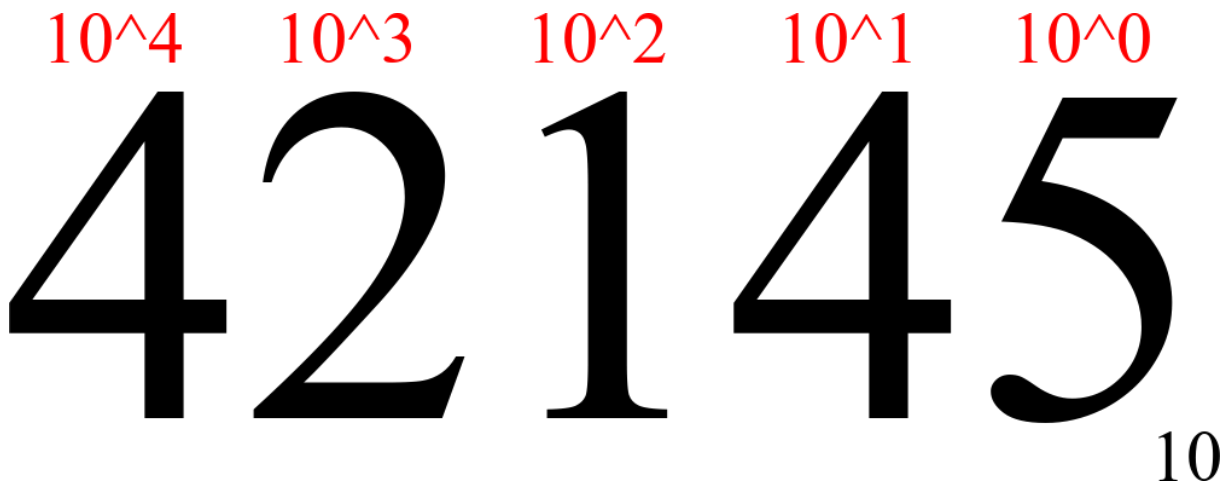
Zadanie 3.2.

Zadanie 3.2. (0–1)

1.	$(10000000)_2$ jest liczbą większą od liczby $(A9)_{16}$	**P**	**F**
2.	$(1111)_4$ jest liczbą większą od liczby $(1111111)_2$	**P**	**F**
3.	$(3003)_4$ jest liczbą większą od liczby $(C2)_{16}$	**P**	**F**
4.	$(333)_8$ jest liczbą większą od liczby $(10100101)_2$	**P**	**F**

Zadanie 3.2 Test z arkusza maturalnego z informatyki maj 2021

Zamiana systemów liczbowych jest typowym zadaniem maturalnym. Każdy system liczbowy ma swoją podstawę. Na co dzień korzystamy z systemu dziesiętnego, którego podstawą są jedności, dziesiątki, setki itd. W rzeczywistości są to potęgi liczby 10: 10^0, 10^1, 10^2…, a więc podstawą jest liczba 10. Podobnie sprawa wygląda przy innych systemach liczbowych, z tym wyjątkiem, że zamiast 10 mamy inną podstawę np. 2, 4, 8 czy 16.



System dziesiętny jako suma potęg.

Konwersja systemu liczbowego na dziesiętny będzie polegać na przemnożeniu liczby przez odpowiednio spotęgowaną podstawę (zależnie od miejsca). Dla przykładu:

- 1032 w systemie czwórkowym to: $2*4^0 + 3*4^1 + 0*4^2 + 1*4^3 = 2*1 + 3*4 + 1*64 = 2 + 12 + 64 = 78$ w systemie dziesiętnym
- 10001 w systemie dwójkowym to: $1*1 + 1*16 = 17$

Posiadając tę wiedzę, możemy przystąpić do rozwiązania zadań:

- $(10000000)_2 = 1*128 = 128$
  $(A9)_{16} = 9*1 + 10*16 = 169$
  Tutaj warto wyjaśnić skąd wziął się znak A w liczbie. W podstawach liczbowych większych od 10 nie istnieje wystarczająca liczba cyfr, aby móc je zapisać, ponieważ jak wspomnieliśmy na co dzień korzystamy z systemu dziesiętnego, w którym występuje 10 cyfr (od 0 do 9). Jak więc zapisać 10 w jednym znaku? Wykorzystano do tego kolejne znaki alfabetu, czyli A, B, C… Każdy znak to kolejna cyfra, czyli A =

10, B = 11, C = 12 itd.
Wracając do zadania, czy liczba 128 jest większa od 169? Ano nie, więc zaznaczamy fałsz.

- $(1111)_4 = 1*1 + 1*4 + 1*16 + 1*64 = 85$
  $(1111111)_2 = 1 + 2 + 4 + 8 + 16 + 32 + 64 = 127$
  Liczba 85 jest większa od 127? Też nie, czyli będzie to fałsz.
- $(3003)_4 = 3*1 + 3*64 = 195$
  $(C2)_{16} = 2*1 + 12*16 = 194$
  Tym razem 195 jest większe od 194, przez co zaznaczamy prawdę.
- $(333)_8 = 3*1 + 3*8 + 3*64 = 219$
  $(10100101)_2 = 1 + 4 + 32 + 128 = 165$
  Tutaj także otrzymujemy prawdę.

Zadanie 3.3.

Zadanie 3.3. (0–1)

W bazie danych istnieje tabela *produkty(id_produktu, produkt, sztuk, cena)*, zawierająca następujące dane:

id_produktu	produkt	sztuk	cena
1	zeszyt	160	2
2	okładka	100	3
3	ołówek	250	1
4	długopis	178	5
5	pióro	100	12
6	gumka	250	1
7	piórnik	125	8
8	cyrkiel	130	4

1.	Wynikiem zapytania **SELECT** produkt **FROM** produkty **WHERE** (cena = 2 **OR** cena = 4) jest cyrkiel	**P**	**F**
2.	Wynikiem zapytania **SELECT** AVG(cena) **FROM** produkty **WHERE** sztuk **IN** (125, 160) jest 5	**P**	**F**
3.	Wynikiem zapytania **SELECT** SUM(sztuk) **FROM** produkty **WHERE** (cena = 1 **OR** cena = 2) jest 660	**P**	**F**
4.	Wynikiem zapytania **SELECT** COUNT(cena) **FROM** produkty **WHERE** cena **BETWEEN** 2 **AND** 4 jest 2	**P**	**F**

Zadanie 3.3 Test z arkusza maturalnego z informatyki maj

Zadania z baz danych wymagają znajomości języka SQL. Każde zapytanie, które wyciąga dane z bazy, rozpoczyna się klauzulą SELECT. Następnie definiujemy, co chcemy otrzymać (a więc wybieramy jedną z kolumn) oraz z jakiej tabeli będziemy wyciągać dane (zapisujemy ją po FROM). W tym przypadku odnosimy się za każdym razem do tabeli produkty, ponieważ tylko ona jest dostępna.
Po tabeli opcjonalnie występuje instrukcja WHERE, która pozwala wybrać konkretne dane, np. produkty, których cena jest mniejsza od 3. Przejdźmy do zadań:

- W tym zapytaniu wybieramy produkty, których cena jest równa 2 lub 4. Dzieje się tak za sprawą spójnika OR. Patrząc na tabelę widzimy, że taką cenę mają dwa produkty: zeszyt oraz cyrkiel, które powinny stanowić odpowiedź. Zaznaczamy fałsz.

- Zapytanie wykorzystuje funkcję agregującą AVG wyznaczającą średnią z ceny przedmiotów, których liczba sztuk wynosi 125 lub 160. Częstym błędem na maturze było potraktowanie zapisu (125, 160) jako zakresu od 125 do 160, co jest fałszywym założeniem. Prawidłowa liczbę sztuk mają produkty zeszyt i piórnik, dlatego liczymy z nich średnią cenę: (2 + 8) / 2 = 5. Jest więc to prawda.
- W kolejnym zapytaniu ponownie pojawia się funkcja agregująca, tym razem SUM, czyli suma. Sumujemy sztuki produktów, których cena jest równa 1 lub 2. Patrzymy więc na zeszyt, ołówek oraz gumkę. Ich łączna liczba sztuk to 160 + 250 + 250 = 660. Zaznaczamy prawdę.
- Ostatni przykład wykorzystuje funkcję agregującą COUNT, oznaczającą zliczaj. Zliczamy liczbę produktów, których cena zawiera się pomiędzy 2 a 4. Produkty z taką ceną to zeszyt, okładka i cyrkiel, co oznacza, że prawidłowa odpowiedź to 3. Zaznaczamy fałsz.

Zadanie 4 – Neon cyfrowy

Zadanie 4. Neon cyfrowy

Pewna firma przygotowuje wyświetlanie napisów złożonych z wielkich liter alfabetu angielskiego. Na początku napis jest pusty (nie zawiera liter). W pliku `instrukcje.txt` podanych jest 2000 instrukcji, które wykonuje automat do generowania napisu. Każda z instrukcji składa się z polecenia, spacji oraz pojedynczego znaku. Polecenia są czterech rodzajów:

DOPISZ *litera* – oznacza, że na końcu napisu trzeba dopisać pojedynczą *literę*;
ZMIEN *litera* – oznacza, że ostatnią literę aktualnego napisu należy zmienić na podaną *literę* (możesz założyć, że napis jest niepusty);
USUN 1 – oznacza, że należy usunąć ostatnią literę aktualnego napisu (możesz założyć, że napis jest niepusty);
PRZESUN *litera* – oznacza, że pierwsze od lewej wystąpienie podanej *litery* w napisie należy zamienić na następną literę w alfabecie (jeśli *litera* to A, to należy zamienić na B, jeśli B, to na C itd.) Literę Z należy zamienić na A. Jeśli *litera* nie występuje w napisie, nie należy nic robić.

Przykład:
Dany jest następujący ciąg instrukcji:
DOPISZ A
DOPISZ B
DOPISZ C
USUN 1
DOPISZ D
ZMIEN B
DOPISZ E
PRZESUN B
Po wykonaniu pierwszych trzech instrukcji napis będzie miał postać ABC, potem AB, ABD, ABB, ABBE, wreszcie ostatnia instrukcja zamieni pierwsze B na C, więc ostatecznie powstały napis to ACBE.

Napisz program (lub kilka programów), który(-e) znajdzie(-dą) odpowiedzi na poniższe pytania. Każdą odpowiedź zapisz w pliku `wyniki4.txt` i poprzedź ją numerem oznaczającym zadanie.
Do dyspozycji masz również plik `przyklad.txt`, w którym znajduje się tylko 20 instrukcji – odpowiedzi dla tego pliku podane są w treściach zadań, możesz więc sprawdzać na nim działanie swojego programu. Pamiętaj, że Twój program musi ostatecznie działać dla 2000 instrukcji.

Zadanie 4 Neon cyfrowy z arkusza maturalnego z informatyki maj

Zadania z programowania są typowym zadaniem egzaminacyjnym. Warto tu pamiętać o przykładowym pliku: pozwala on sprawdzać, czy nasz kod zwraca poprawne wyniki.Zacznijmy od odczytania danych z dostarczonego pliku. Wykorzystamy do tego instrukcję **with open(nazwa_pliku) as nazwa**:

```
1instrukcje = []
```

```
with open("Dane_2105/instrukcje.txt") as plik:
    for linia in plik:
        instrukcja = linia.strip() # czyszczenie linii z białych znaków
        instrukcje.append(instrukcja)

```

Tworzymy listę instrukcje, w której po kolei zapiszemy na później instrukcje z pliku. Otwieramy w tym celu plik, po którym iterujemy linia po linii (**for linia in plik**).
Później należy oczyścić linię z białych znaków (takich jak znak nowej linii czy spacja), aby później praca była bezproblemowa. Ostatecznie dodajemy instrukcję do stworzonej wcześniej listy.

Zadanie 4.1.

Zadanie 4.1. (0–2)

Oblicz całkowitą długość napisu po wykonaniu wszystkich instrukcji z pliku `instrukcje.txt`.

Dla pliku `przyklad.txt` długością napisu jest liczba 10.

Zadanie 4.1 Neon cyfrowy z arkusza maturalnego z informatyki maj

```
dlugosc = 0
for ins in instrukcje:
    if ins.startswith("DOPISZ"):
        dlugosc += 1
    elif ins.startswith("USUN"):
        dlugosc -= 1

print("1)", dlugosc)
```

Aby rozwiązać to zadanie, musimy zacząć od utworzenia zmiennej służącej do przechowywania długości napisu. Następnie dla każdej instrukcji DOPISZ dodajemy do tej zmiennej 1, a dla USUN usuwamy. Wykorzystujemy tu metodę **startswith**, która jest dostępna dla każdego ciągu znaków. Jak sama nazwa sugeruje, sprawdza ona, czy dany ciąg znaków zaczyna się od podanego napisu.
Reszta instrukcji nas nie interesuje, ponieważ nie zmieniają one długości napisu.
Na koniec wypisujemy długość napisu, aby móc ją zapisać do pliku wyniki4.txt.

Zadanie 4.2.

Zadanie 4.2. (0–2)

Znajdź najdłuższy ciąg występujących kolejno po sobie instrukcji tego samego rodzaju. Jako odpowiedź podaj rodzaj instrukcji oraz długość tego ciągu. Istnieje tylko jeden taki ciąg.

Dla pliku `przyklad.txt` odpowiedzią jest: rodzaj instrukcji – DOPISZ, długość ciągu – 5.

Zadanie 4.2 Neon cyfrowy z arkusza maturalnego z informatyki maj

W tym zadaniu będziemy potrzebowali kilku zmiennych. Przede wszystkim musimy mieć dwie zmienne, w których zapiszemy rodzaj instrukcji, które występują aktualnie w ciągu oraz

ich długość. Oprócz tego potrzebujemy miejsce do zapisania najdłuższego ciągu, który wystąpił do tej pory (wraz z rodzajem instrukcji):

```
1
2
3  aktualny_rodzaj = None
4  aktualna_dlugosc = 0
5  najdluzszy_rodzaj = None
6  najdluzsza_dlugosc = 0
7  for ins in instrukcje:
8      rodzaj = ins.split()[0]
9      if not aktualny_rodzaj:
10         aktualny_rodzaj = rodzaj
11         aktualna_dlugosc = 1
12         continue
13     if aktualny_rodzaj == rodzaj:
14         aktualna_dlugosc += 1
15         if aktualna_dlugosc > najdluzsza_dlugosc:
16             # ten ciąg jest w tym momencie najdłuższy
17             najdluzsza_dlugosc = aktualna_dlugosc
18             najdluzszy_rodzaj = rodzaj
19     else:
20         aktualny_rodzaj = rodzaj
21         aktualna_dlugosc = 1

print("2)", najdluzszy_rodzaj, najdluzsza_dlugosc)
```

Kolejny raz korzystamy z pętli for, w której iterujemy po instrukcjach. Tym razem liczy się tylko rodzaj instrukcji (ZMIEN, USUN…), co uzyskujemy instrukcją **ins.split()[0]**.
Wykonuje ona podział elementów oddzielonych białymi znakami (w naszym przypadku spacją) i tworzy z tych elementów listę. Dla przykładu: **‘DOPISZ A’.split()** zwróci nam listę [‘DOPISZ’, ‘A’], a więc rodzaj instrukcji zawiera się pod zerowym indeksem.
Następnie sprawdzamy, czy cokolwiek znajduje się pod zmienną **aktualny_rodzaj**. Taki przypadek ma miejsce tylko podczas pierwszej iteracji pętli, jako że przypisaliśmy aktualnemu rodzajowi wartość **None**. Po przypisaniu rodzaju i długości używamy instrukcji **continue**, która przechodzi do kolejnej iteracji pętli (a więc pomija kroki znajdujące się poniżej).
Jeśli zmienna **aktualny_rodzaj** jest już określona, sprawdzamy czy kolejna instrukcja jest tego samego rodzaju.
Jeśli tak, musimy zwiększyć długość ciągu o jeden i sprawdzić, czy aktualny ciąg jest najdłuższy.
Jeśli nie, oznacza to, że rozpoczyna się nowy ciąg, który przyjmuje początkową długość 1, jako że do tej pory tylko raz wystąpiła instrukcja tego samego typu.

W ten sposób uzyskujemy rodzaj i długość najdłuższego ciągu, którą wypisujemy.

Zadanie 4.3.

Zadanie 4.3. (0–3)

Oblicz, która litera jest najczęściej dopisywana (najczęściej występuje w instrukcji DOPISZ). Podaj tę literę oraz ile razy jest dopisywana. Istnieje tylko jedna taka litera.

Dla pliku `przyklad.txt` odpowiedzią jest litera U, dopisywana 3 razy.

Zadanie 4.3 Neon cyfrowy z arkusza maturalnego z informatyki maj

W tym zadaniu przydatna jest znajomość Pythonowych słowników (zwanych też tablicą asocjacyjną). Słownik jest strukturą, w której dane zapisane są w formacie klucz:wartość. W tym przypadku kluczem będzie litera, a wartością liczba wystąpień tej litery. Aby odwołać się do wartości, zapisujemy litery[litera]. **Wiedza ta jest przydatna, ale nie obowiązkowa. Zadanie można bez większego problemu rozwiązać bez użycia słowników!** My jednak przedstawimy rozwiązanie wykorzystujące słowniki.

```
1
2 litery = {}
3 for ins in instrukcje:
4     rodzaj, znak = ins.split()
5     if rodzaj == "DOPISZ":
6         if znak in litery:
7             litery[znak] += 1
8         else:
9             litery[znak] = 1
10
11najczesciej_litera = None
12najczesciej = 0
13for litera in litery:
14    if litery[litera] > najczesciej:
15        najczesciej_litera = litera
16        najczesciej = litery[litera]
17# skrócona wersja powyższego dla ciekawych:
18# najczesciej = max(litery.items(), key=lambda x: x[1])
19print("3)", najczesciej_litera, najczesciej)
20
```

Dla każdej instrukcji sprawdzamy, czy dotyczy ona dopisania. Następnie patrzymy, czy dany znak znajduje się w słowniku litery.
Jeśli tak, to dodajemy 1 do liczby wystąpień tej litery.
Jeśli nie, to tworzymy nowy wpis w słowniku z wartością 1.
W ten sposób otrzymujemy ile razy każdy znak został dopisany. Wystarczy teraz znaleźć maksymalną liczbę, co robimy za pomocą pętli for.

Zadanie 4.4.

Zadanie 4.4 (0–4)

Podaj napis, który powstanie po wykonaniu wszystkich instrukcji z pliku `instrukcje.txt`.

Dla pliku `przyklad.txt` wynikiem jest napis ALANTURING.

Zadanie 4.4 Neon cyfrowy z arkusza maturalnego z informatyki maj

W tym zadaniu wykorzystamy wiedzę z poprzednich poleceń.
Opiszmy każdą instrukcję po kolei:

a) DOPISZ: dopisujemy do napisu podany znak, a więc możemy to zapisać w formie **napis += znak** (co jest równoznaczne z zapisem **napis = napis + znak**).

b) ZMIEN: zmieniamy ostatni znak na inny. Aby to zrobić, musimy najpierw pozbyć się ostatniej litery, a następnie dodać podaną. Korzystamy w tym celu z tzw. wycinania (ang. slicing), za pomocą którego wybierzemy znaki znajdujące się od początku aż do przedostatniego znaku w napisie. Zapisujemy to w ten sposób: **napis[:-1]**. W ten sposób usunęliśmy ostatni znak, teraz zostaje dodać nowy**: napis = napis[:-1] + znak**.

c) USUN: tutaj także skorzystamy z wykrajania, jako że ponownie musimy pozbyć się ostatniego wyrazu: **napis = napis[:-1]**.

d) PRZESUN: w tej instrukcji mamy najwięcej do zrobienia. Do zamienienia pierwszego wystąpienia od lewej możemy wykorzystać funkcję **napis.replace**, w którym jako argumenty podajemy znak do zmienienia, nowy znak oraz opcjonalnie liczbę zmian do wykonania. Tutaj chcemy dokonać tylko jednej zmiany, dlatego w ostatnim argumencie zapisujemy 1. Będzie to wyglądać w ten sposób: **napis = napis.replace(znak, nowy_znak, 1)**.

Musimy teraz w jakiś sposób wyznaczyć ten nowy znak. Zgodnie z podaną w zadaniu definicją, zamieniamy literę na kolejną w alfabecie. W tym celu możemy skorzystać z kodów ASCII, w których każdy znak ma określony kod liczbowy. Duże litery alfabetu zaczynają się od 65 i kończą na 90 (A to 65, a Z to 90), co wykorzystamy w mechanizmie zamiany. Aby w Pythonie uzyskać kod ASCII znaku, przekazujemy znak do funkcji **ord**, czyli kod_ascii = ord(znak). Następną literę w alfabecie uzyskujemy poprzez dodanie do tego kodu 1 i zamienienie go z powrotem na typ znakowy z użyciem funkcji **chr** (od character, po polsku znak): **nowy_znak = chr(kod_ascii+1)**.
Wyjątek od tej reguły stanowi znak Z, jako że gdy dodamy do kodu ASCII tej litery 1, otrzymamy wynik 91, który oznacza znak [. Najłatwiejszym sposobem na rozwiązanie tego problemu jest wcześniejsze sprawdzenie, czy znak do zmiany to Z i jeśli tak, to nowym znakiem będzie A:

```
1 if znak == "Z":
2     nowy_znak = "A"
3 else:
4     kod_ascii = ord(znak)
5     nowy_znak = chr(kod_ascii+1) napis =
6 napis.replace(znak, nowy_znak, 1)
```

Pełne rozwiązanie tego zadania prezentuje się w ten sposób:

```
1  napis = ""
2  for ins in instrukcje:
3      rodzaj, znak = ins.split()
4      if rodzaj == "DOPISZ":
5          napis += znak
6      elif rodzaj == "ZMIEN":
7          napis = napis[:-1] + znak
8      elif rodzaj == "USUN":
9          napis = napis[:-1]
```

```
    elif rodzaj == "PRZESUN":
        if znak == "Z":
            nowy_znak = "A"
        else:
            kod_ascii = ord(znak)
            nowy_znak = chr(kod_ascii+1)
        napis = napis.replace(znak, nowy_znak, 1)

print("4)", napis)
```

Zadanie 5 – Wodociągi

Zadanie 5. Wodociągi

Wodociągi miejskie zamierzają wykonać analizę zużycia wody. W tym celu zgromadziły dane o poborze wody przez wszystkich swoich klientów za rok 2019. Dane są zapisane w pliku `wodociagi.txt`. Pierwszy wiersz pliku jest wierszem nagłówkowym, a dane rozdzielono średnikami. W każdym wierszu zapisano informacje dotyczące gospodarstwa domowego jednego klienta: dziesięcioznakowy kod klienta oraz 12 liczb całkowitych oznaczających ilości zużytej wody w m^3 przez kolejnych 12 miesięcy (od stycznia do grudnia). Kod klienta składa się z pięciocyfrowego numeru klienta, dwucyfrowej liczby oznaczającej liczbę osób pozostających we wspólnym gospodarstwie domowym oraz trzyliterowego kodu dzielnicy miasta. Każdy kod jest unikatowy.

Fragment pliku `wodociagi.txt`:

```
KodKlienta;I;II;III;IV;V;VI;VII;VIII;IX;X;XI;XII
0000103WIL;6;6;6;9;6;15;12;12;12;6;9;6
0000403BEM;6;3;9;9;12;15;15;15;9;6;3;9
```

Korzystając z powyższych danych oraz dostępnych narzędzi informatycznych, wykonaj podane zadania. Wyniki zapisz w pliku tekstowym `wyniki5.txt`. Odpowiedź do każdego zadania poprzedź numerem tego zadania.

Zadanie 5 Wodociągi z arkusza maturalnego z informatyki maj

Do rozwiązania tego zadania wykorzystamy arkusz kalkulacyjny. Zaczynamy od importu danych, który w Excelu znajduje się pod pozycją Dane -> Z pliku tekstowego. Wybieramy dostarczony plik i bez większych zmian importujemy do arkusza.

Zadanie 5.1.

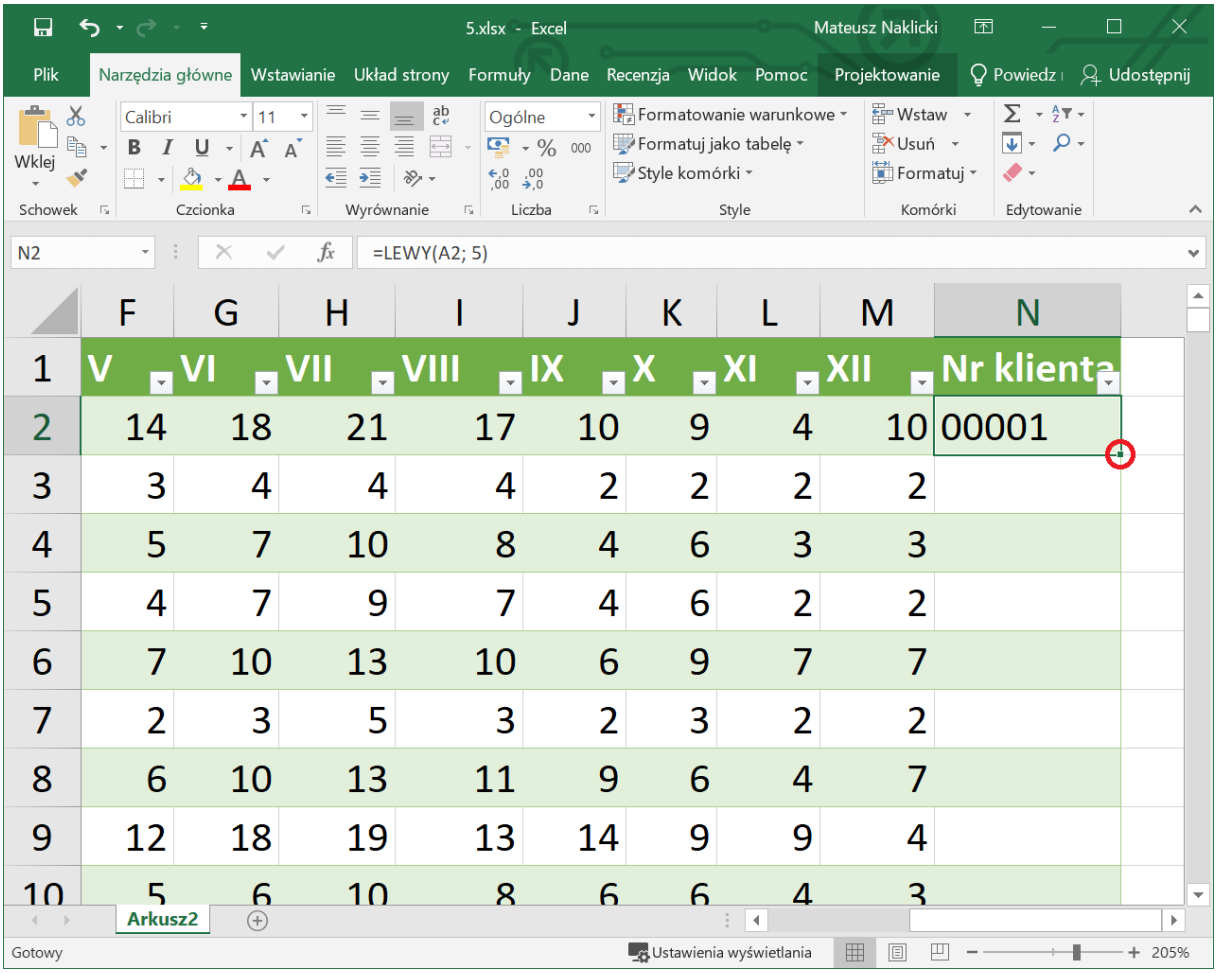
Zadanie 5.1. (0–3)

Utwórz zestawienie zawierające **pięciocyfrowe numery** 10 klientów, którzy w ciągu roku zużyli w swoim gospodarstwie domowym średnio najwięcej wody na jedną osobę, oraz ich średnie zużycie wody na jedną osobę. Średnioroczne zużycie wody na jedną osobę zaokrąglij do dwóch miejsc po przecinku.

Zestawienie, zawierające numery klientów i średnie zużycie wody na jedną osobę, uporządkuj nierosnąco według średniej.

Zadanie 5.1 Wodociągi z arkusza maturalnego z informatyki maj

Rozpoczniemy od przygotowania pięciocyfrowych numerów klientów, które znajdują się na początku kolumny KodKlienta. Aby wydobyć 5 pierwszych znaków, skorzystamy z formuły LEWY, gdzie jako pierwszy argument podajemy pole z kolumny KodKlienta, a jako drugi liczbę znaków, którą chcemy uzyskać: w naszym wypadku 5.
Finalnie instrukcja powinna wyglądać w ten sposób: **=LEWY(A2; 5)**. Klikamy teraz dwukrotnie w prawy dolny róg komórki, aby wykonać to polecenie dla każdego wiersza:



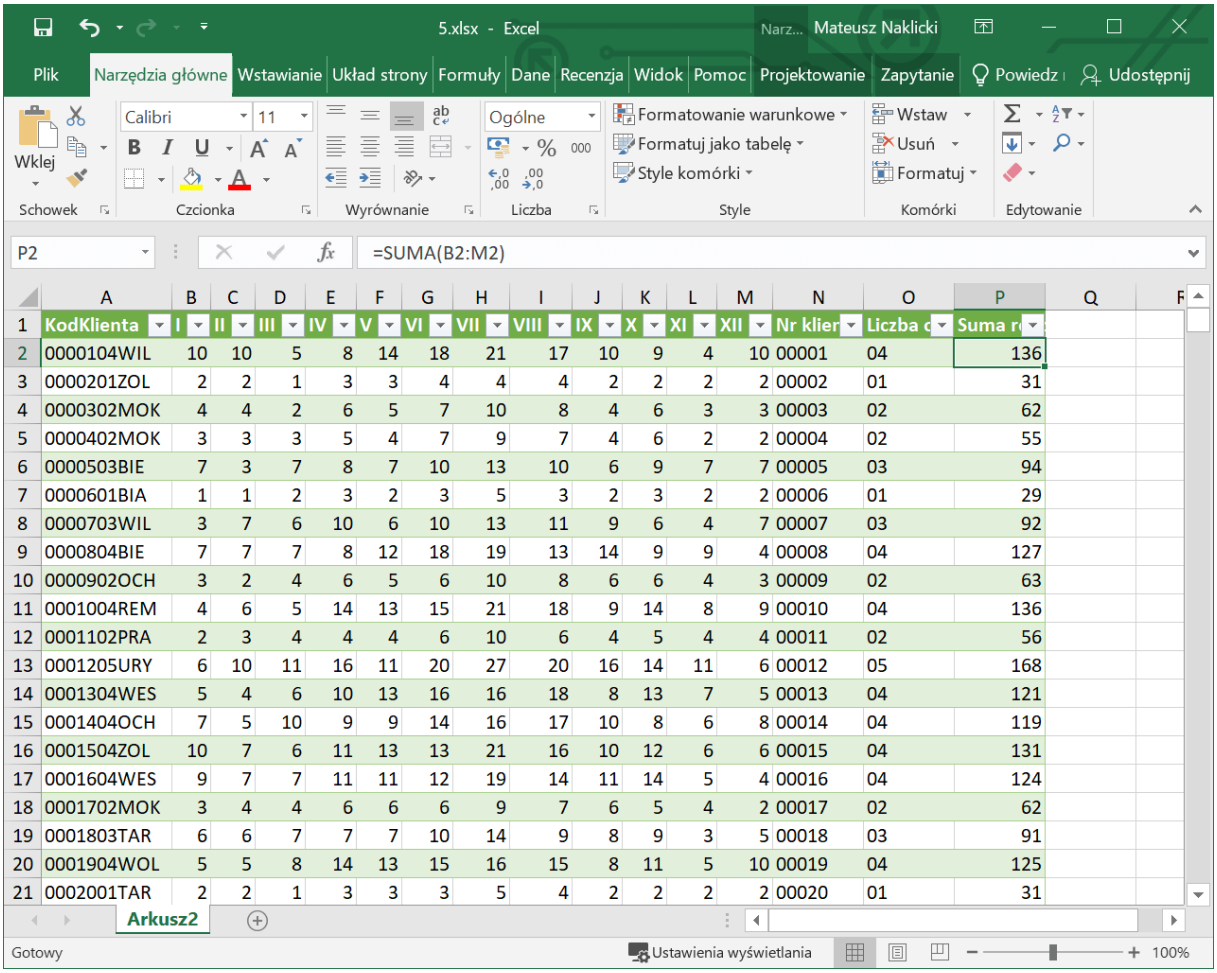
	F	G	H	I	J	K	L	M	N
1	V	VI	VII	VIII	IX	X	XI	XII	Nr klienta
2	14	18	21	17	10	9	4	10	00001
3	3	4	4	4	2	2	2	2	
4	5	7	10	8	4	6	3	3	
5	4	7	9	7	4	6	2	2	
6	7	10	13	10	6	9	7	7	
7	2	3	5	3	2	3	2	2	
8	6	10	13	11	9	6	4	7	
9	12	18	19	13	14	9	9	4	
10	5	6	10	8	6	6	4	3	

Instrukcja =LEWY(A2; 5).

Teraz potrzebujemy liczby osób w gospodarstwie, która także znajduje się w kodzie klienta. Są to dwie cyfry po numerze klienta. Aby je wydobyć, skorzystamy zarówno z funkcji LEWY, jak i PRAWY: najpierw wydobędziemy pierwsze 7 znaków, a z wydobytego tekstu dwa ostatnie: **=PRAWY(LEWY(A2;7); 2)**.

Zsumujmy też zużycie wody z całego roku w osobnej kolumnie, korzystając z formuły SUMA i podając jako zakres wszystkie miesiące: **=SUMA(B2:M2)**.

Tabela po tych operacjach powinna wyglądać w ten sposób:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	KodKlienta	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	Nr klien	Liczba d	Suma r
2	0000104WIL	10	10	5	8	14	18	21	17	10	9	4	10	00001	04	136
3	0000201ZOL	2	2	1	3	3	4	4	4	2	2	2	2	00002	01	31
4	0000302MOK	4	4	2	6	5	7	10	8	4	6	3	3	00003	02	62
5	0000402MOK	3	3	3	5	4	7	9	7	4	6	2	2	00004	02	55
6	0000503BIE	7	3	7	8	7	10	13	10	6	9	7	7	00005	03	94
7	0000601BIA	1	1	2	3	2	3	5	3	2	3	2	2	00006	01	29
8	0000703WIL	3	7	6	10	6	10	13	11	9	6	4	7	00007	03	92
9	0000804BIE	7	7	7	8	12	18	19	13	14	9	9	4	00008	04	127
10	0000902OCH	3	2	4	6	5	6	10	8	6	6	4	3	00009	02	63
11	0001004REM	4	6	5	14	13	15	21	18	9	14	8	9	00010	04	136
12	0001102PRA	2	3	4	4	4	6	10	6	4	5	4	4	00011	02	56
13	0001205URY	6	10	11	16	11	20	27	20	16	14	11	6	00012	05	168
14	0001304WES	5	4	6	10	13	16	16	18	8	13	7	5	00013	04	121
15	0001404OCH	7	5	10	9	9	14	16	17	10	8	6	8	00014	04	119
16	0001504ZOL	10	7	6	11	13	13	21	16	10	12	6	6	00015	04	131
17	0001604WES	9	7	7	11	11	12	19	14	11	14	5	4	00016	04	124
18	0001702MOK	3	4	4	6	6	6	9	7	6	5	4	2	00017	02	62
19	0001803TAR	6	6	7	7	7	10	14	9	8	9	3	5	00018	03	91
20	0001904WOL	5	5	8	14	13	15	16	15	8	11	5	10	00019	04	125
21	0002001TAR	2	2	1	3	3	3	5	4	2	2	2	2	00020	01	31

Tabela po wykonaniu powyższych operacji.

Przed utworzeniem zestawienia przygotujmy jeszcze wymagane średnie zużycie wody na jedną osobę. Wystarczy podzielić wyznaczoną sumę przez liczbę osób: przy takim układzie kolumn jak na zrzucie, formuła będzie wyglądać tak: **=P2/O2**. Zgodnie z poleceniem wynik ten należy jeszcze zaokrąglić do dwóch miejsc po przecinku. Służy do tego formuła ZAOKR, w której podajemy liczbę do zaokrąglenia i liczbę miejsc po przecinku. Ostatecznie wygląda to tak: **=ZAOKR(P3/O3;2)**

Aby wybrać 10 klientów, którzy zużyli najwięcej wody na osobę, musimy posortować dane. W tym celu stwórzmy nową tabelę przestawną, którą można znaleźć w zakładce Wstawianie -> Tabele -> Tabela przestawna:

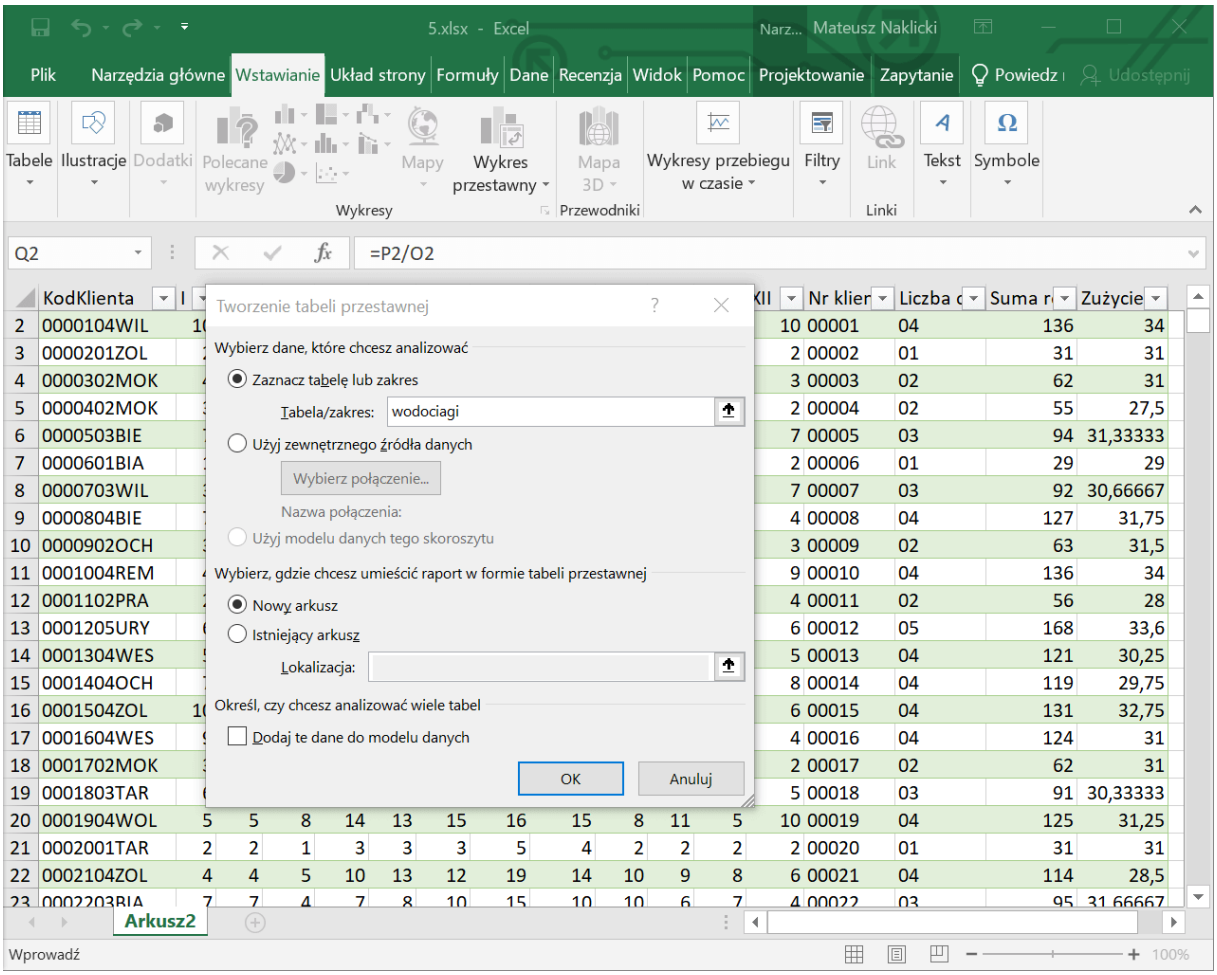
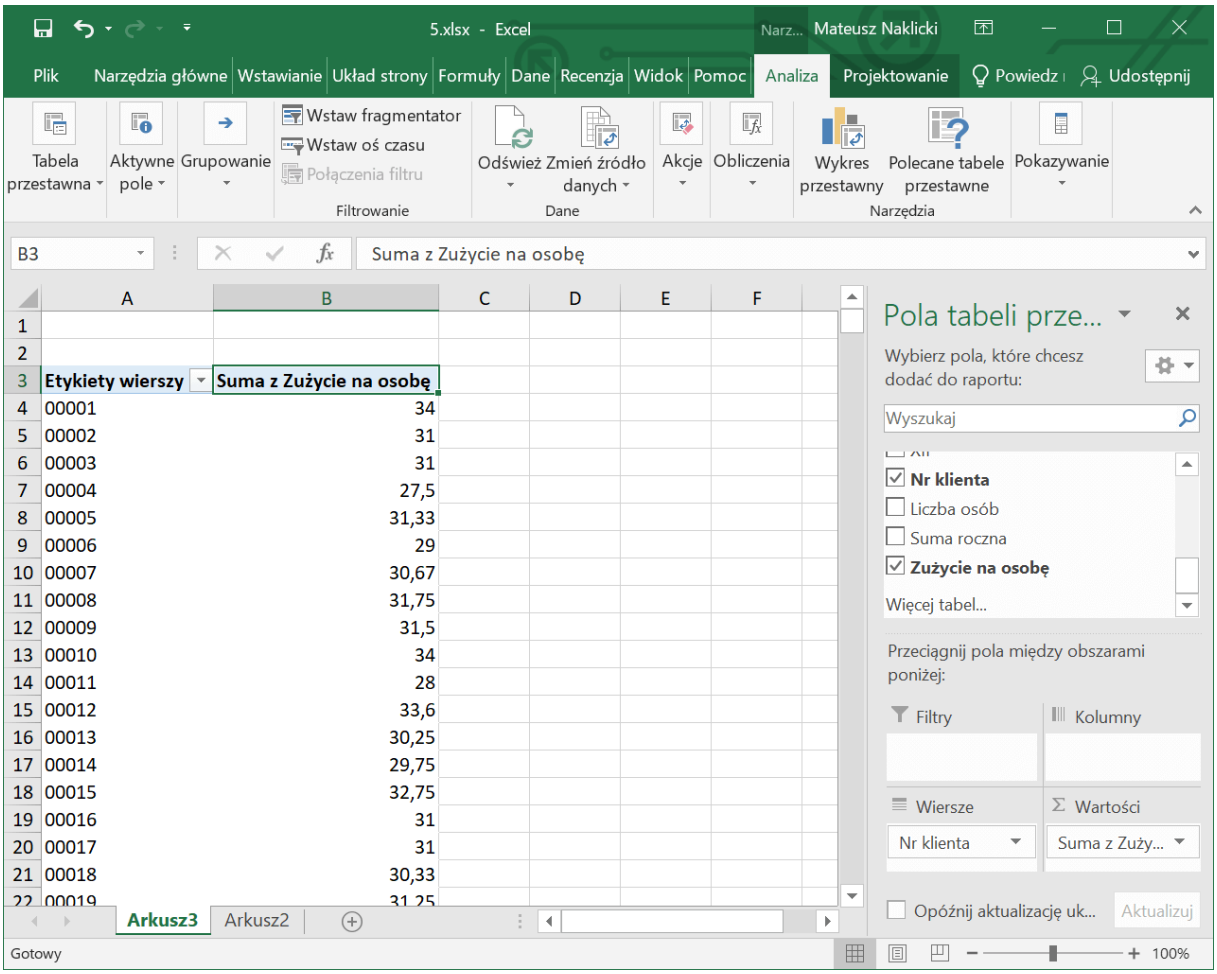


Tabela przestawna.

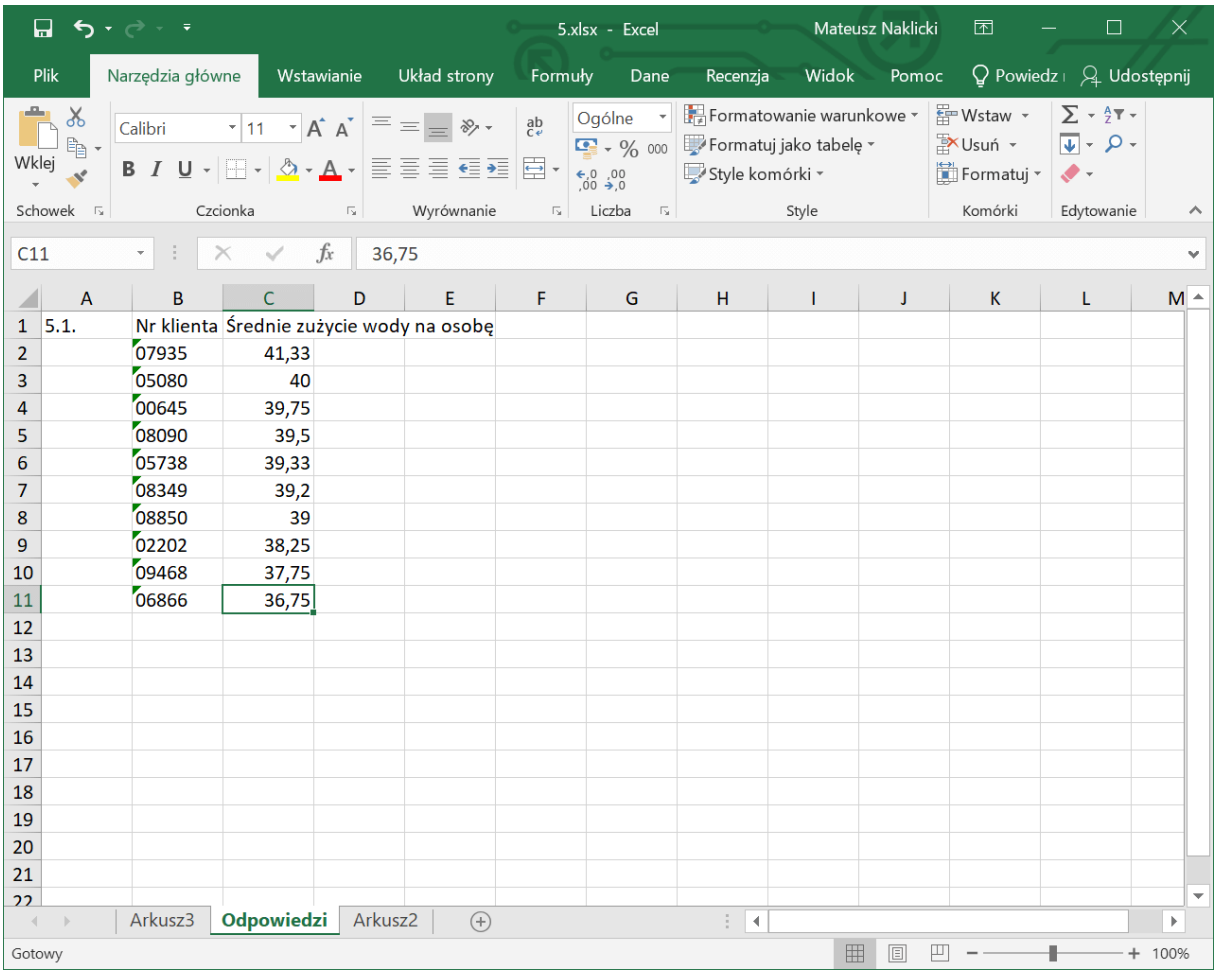
W panelu tabeli przestawnej wybieramy pola Nr klienta i Zużycie na osobę (w zależności od nazw, jakie wcześniej podaliśmy):



Wybieramy pola.

Aby posortować dane, klikamy na jakąkolwiek wartość w kolumnie zawierającą średnie zużycie prawym przyciskiem myszy i wybieramy Sortuj -> Sortuj od największych do najmniejszych.

Wystarczy teraz skopiować 10 pierwszych wpisów i podać je jako odpowiedź do zadania. Możemy w tym celu stworzyć specjalny arkusz o nazwie odpowiedzi:



Tworzymy arkusz odpowiedzi.

Nie zapomnijmy jednak też o skopiowaniu wyników do pliku wyniki6.txt.

Zadanie 5.2.

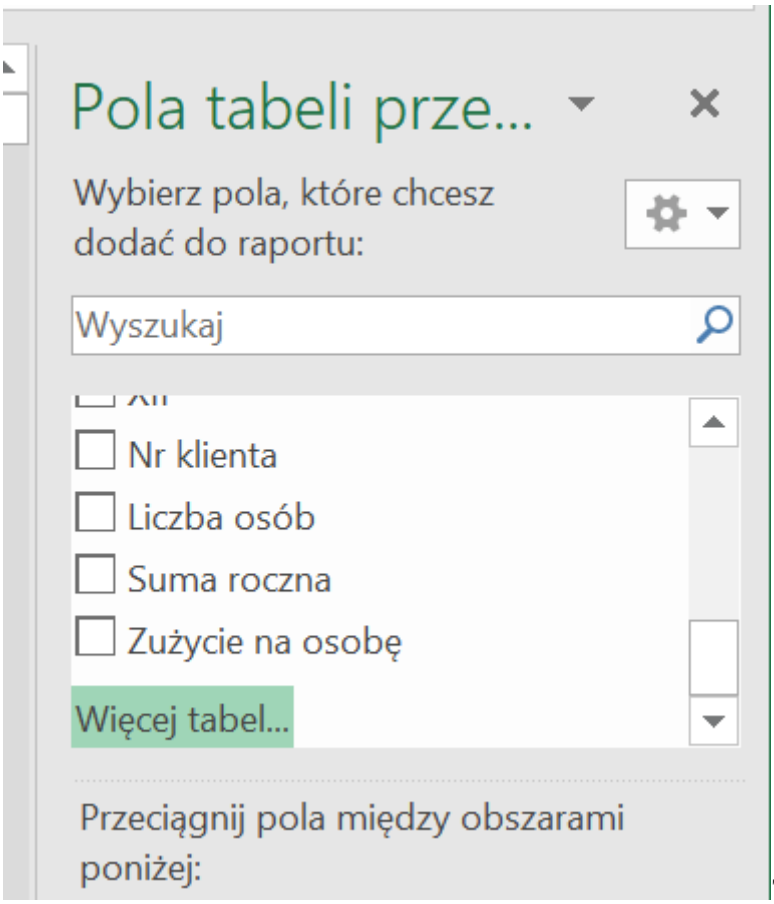
Zadanie 5.2. (0–2)

Dla każdej dzielnicy podaj całkowite roczne zużycie wody przez jej wszystkich mieszkańców.

Zadanie 5.2 Wodociągi z arkusza maturalnego z informatyki maj

Jako że potrzebujemy kodów dzielnic, zaczniemy od ich pozyskania. One także zapisane są w kodzie klienta, lecz tym razem na końcu. Skorzystamy z formuły PRAWY: **=PRAWY(A2;3)**.

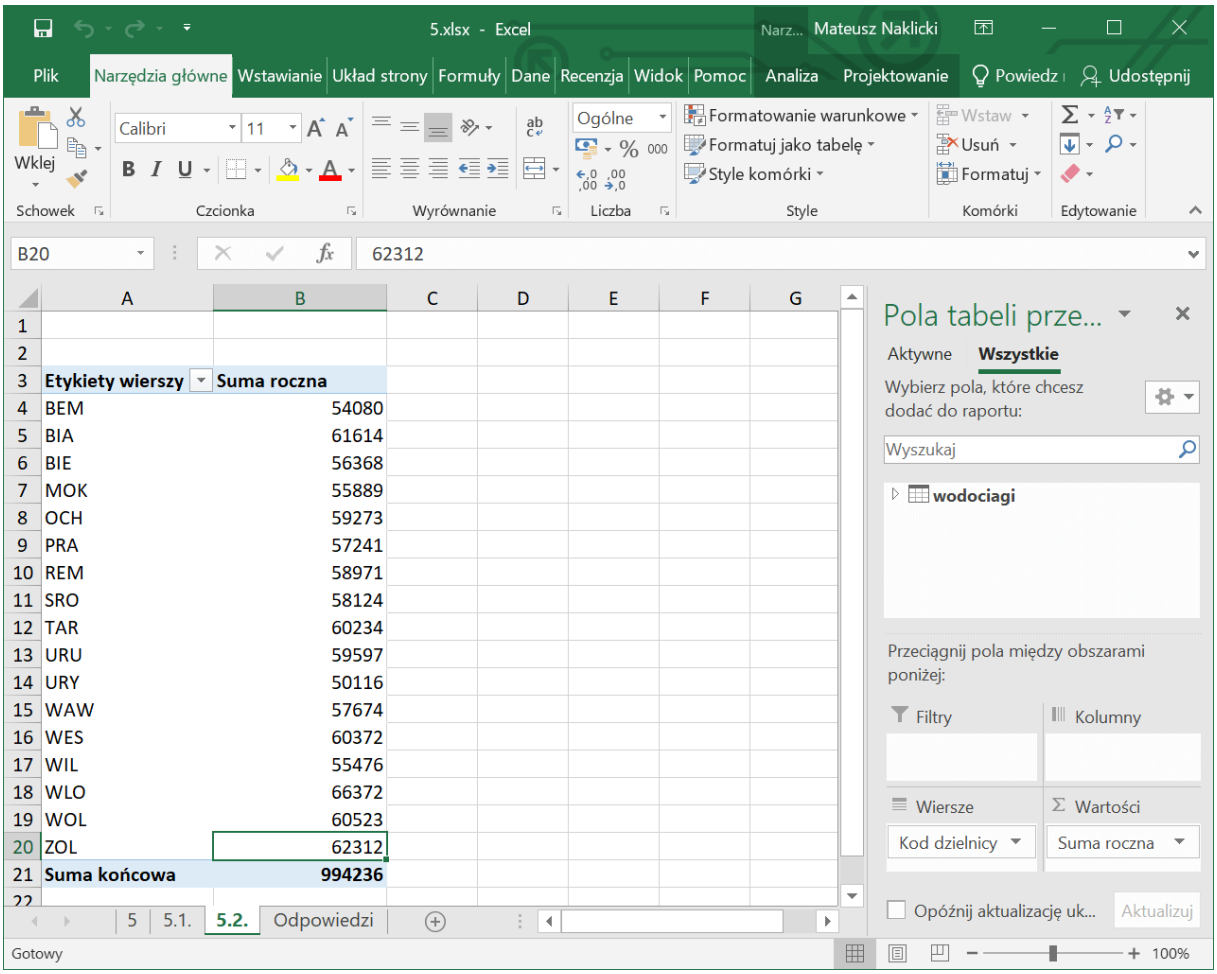
Do tego zadania także wykorzystamy tabelę przestawną. Tworzymy ją w ten sam sposób jak wcześniej. Z niewiadomych mi przyczyn nowo utworzona kolumna Kod dzielnicy nie chciała pojawić się w polach do wyboru tabeli przestawnej. Jeśli u was zdarzyło się coś podobnego, możecie spróbować kliknąć przycisk Więcej tabel na końcu pól do wyboru:



Tworzymy tabelę przestawną.

Lista pól do wyboru powinna się odświeżyć.

Wybieramy Kod dzielnicy jako etykiety wierszy oraz sumę roczną jako wartości. Powinno pojawić się nam ładne zestawienie zużycia wody w każdej dzielnicy:



Wybieramy Kod dzielnicy jako etykiety wierszy oraz sumę roczną jako wartości.

Możemy przenieść zestawienie do arkusza odpowiedzi.

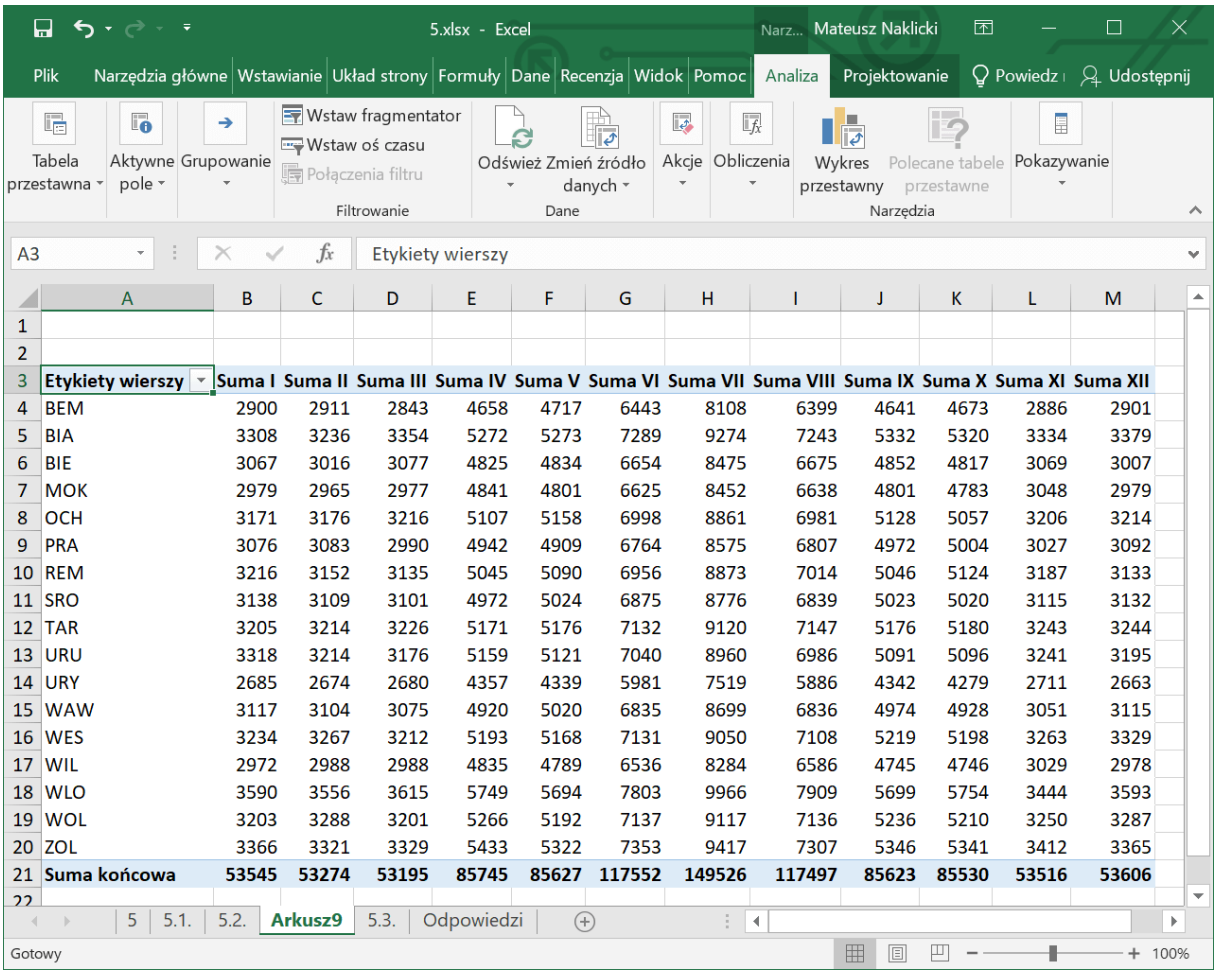
Zadanie 5.3.

Zadanie 5.3. (0–2)

Dla każdej dzielnicy oblicz zużycie wody w każdym miesiącu łącznie przez wszystkich mieszkańców tej dzielnicy. Podaj maksymalne miesięczne zużycie wody w każdej z dzielnic.

Zadanie 5.3 Wodociągi z arkusza maturalnego z informatyki maj

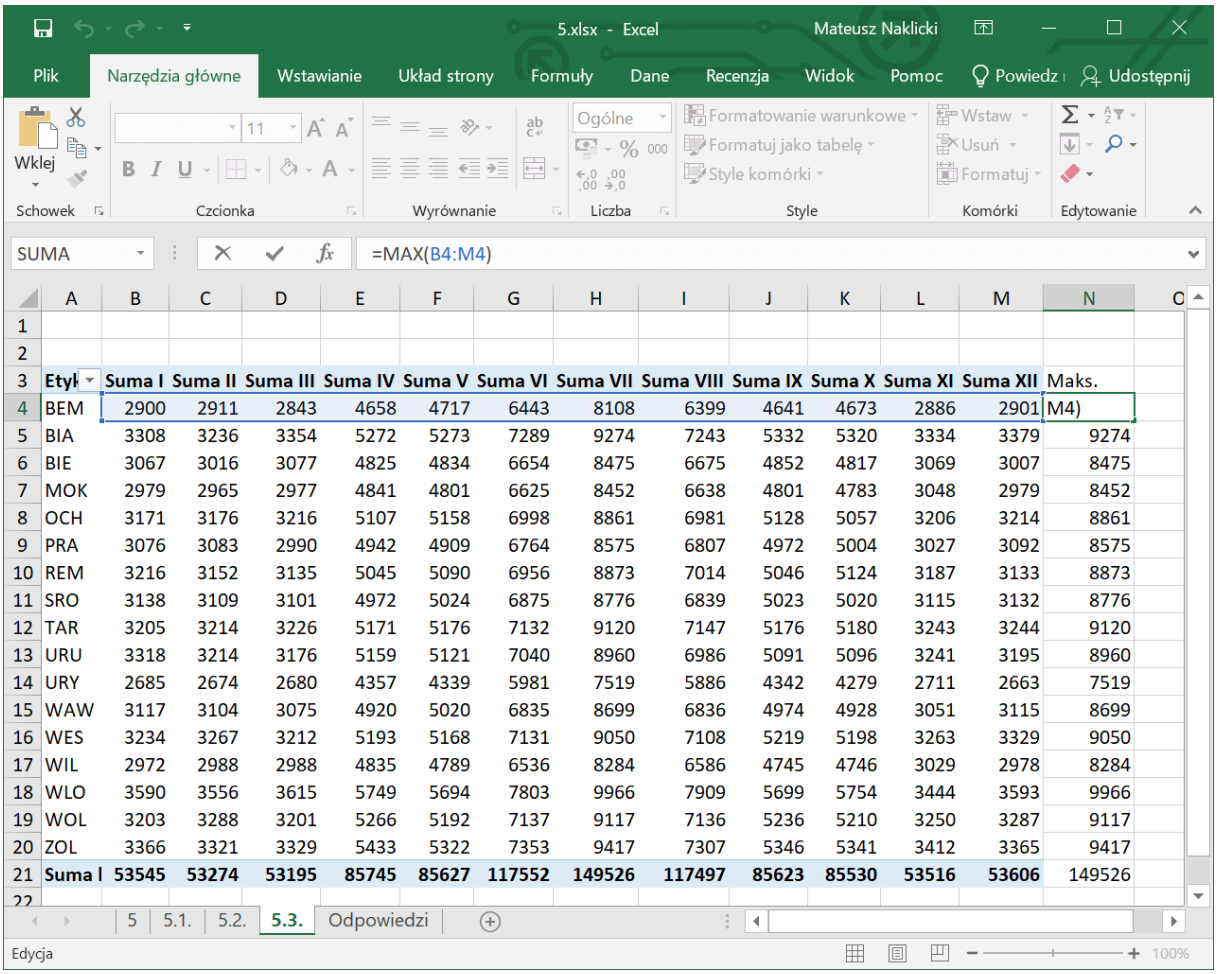
Zadanie jest podobne do wcześniejszego, z tą różnicą, że tu musimy podać zużycie dla każdego miesiąca. Stwórzmy nową tabelę przestawną, w której powiążemy dzielnice z miesiącami:



Etykiety wierszy	Suma I	Suma II	Suma III	Suma IV	Suma V	Suma VI	Suma VII	Suma VIII	Suma IX	Suma X	Suma XI	Suma XII
BEM	2900	2911	2843	4658	4717	6443	8108	6399	4641	4673	2886	2901
BIA	3308	3236	3354	5272	5273	7289	9274	7243	5332	5320	3334	3379
BIE	3067	3016	3077	4825	4834	6654	8475	6675	4852	4817	3069	3007
MOK	2979	2965	2977	4841	4801	6625	8452	6638	4801	4783	3048	2979
OCH	3171	3176	3216	5107	5158	6998	8861	6981	5128	5057	3206	3214
PRA	3076	3083	2990	4942	4909	6764	8575	6807	4972	5004	3027	3092
REM	3216	3152	3135	5045	5090	6956	8873	7014	5046	5124	3187	3133
SRO	3138	3109	3101	4972	5024	6875	8776	6839	5023	5020	3115	3132
TAR	3205	3214	3226	5171	5176	7132	9120	7147	5176	5180	3243	3244
URU	3318	3214	3176	5159	5121	7040	8960	6986	5091	5096	3241	3195
URY	2685	2674	2680	4357	4339	5981	7519	5886	4342	4279	2711	2663
WAW	3117	3104	3075	4920	5020	6835	8699	6836	4974	4928	3051	3115
WES	3234	3267	3212	5193	5168	7131	9050	7108	5219	5198	3263	3329
WIL	2972	2988	2988	4835	4789	6536	8284	6586	4745	4746	3029	2978
WLO	3590	3556	3615	5749	5694	7803	9966	7909	5699	5754	3444	3593
WOL	3203	3288	3201	5266	5192	7137	9117	7136	5236	5210	3250	3287
ZOL	3366	3321	3329	5433	5322	7353	9417	7307	5346	5341	3412	3365
Suma końcowa	**53545**	**53274**	**53195**	**85745**	**85627**	**117552**	**149526**	**117497**	**85623**	**85530**	**53516**	**53606**

W tabeli przestawnej wiążemy dzielnice z miesiącami.

Teraz musimy znaleźć maksymalne wartości dla każdej dzielnicy. W kolumnie obok zapisujemy więc formułę **MAX**, w której zaznaczamy wszystkie miesiące dla każdej dzielnicy:



Wpisujemy formułę MAX.

Teraz przenosimy wyniki (kod dzielnicy wraz z maksymalnym zużyciem) do arkusza odpowiedzi. Jeśli pojawia się problem z adresami komórek przy kopiowaniu, najłatwiej go rozwiązać wklejając dane do np. notatnika, a stamtąd skopiować do arkusza.

Zadanie 5.4.

Zadanie 5.4. (0–4)

Dział inwestycji analizuje konieczność modernizacji sieci wodociągowej na podstawie danych za rok 2019. Jako podstawę obliczeń bierze sumaryczne zużycie wody w każdym z 12 miesięcy. Inżynierowie założyli, że sumaryczne miesięczne zużycie wody będzie rosło o 1% rok do roku każdego miesiąca (w m^3 z zaokrągleniem w górę do najbliższej liczby całkowitej).

Przykład:
jeśli w styczniu 2019 roku sumaryczne zużycie wody w mieście wyniosło 53 545 m^3, to w styczniu 2020 przewidywane zużycie wyniesie 54 081 m^3.

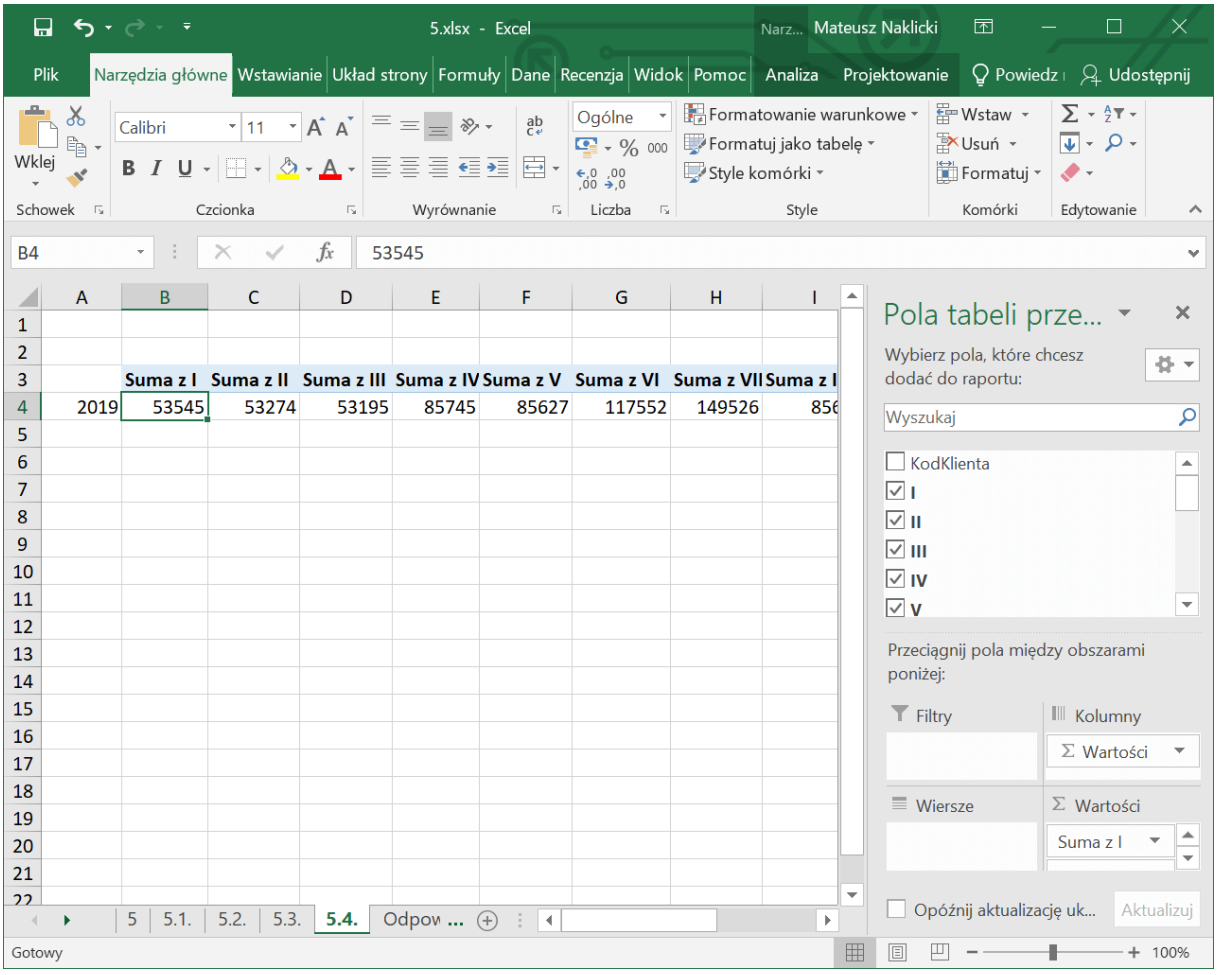
Uwaga: dla danych z zadania przewidywane zużycie wody w maju 2025 roku wyniesie 90 898 m^3.

Zadanie 5.4 Wodociągi z arkusza maturalnego z informatyki maj – część pierwsza

Obecnie maksymalny miesięczny przepływ (wydajność sieci) wynosi 160 000 m^3. Podaj rok i miesiąc, w którym pierwszy raz zabraknie wody w mieście (przewidywane zużycie będzie większe niż maksymalny przepływ sieci).
Sporządź zestawienie obrazujące przewidywane zużycie wody w każdym z kolejnych miesięcy od stycznia 2020 roku do grudnia 2030 roku.
Narysuj wykres liniowy obrazujący przewidywane zużycie wody w każdym z kolejnych miesięcy w **2030 roku**.

Zadanie 5.4 Wodociągi z arkusza maturalnego z informatyki maj – część druga

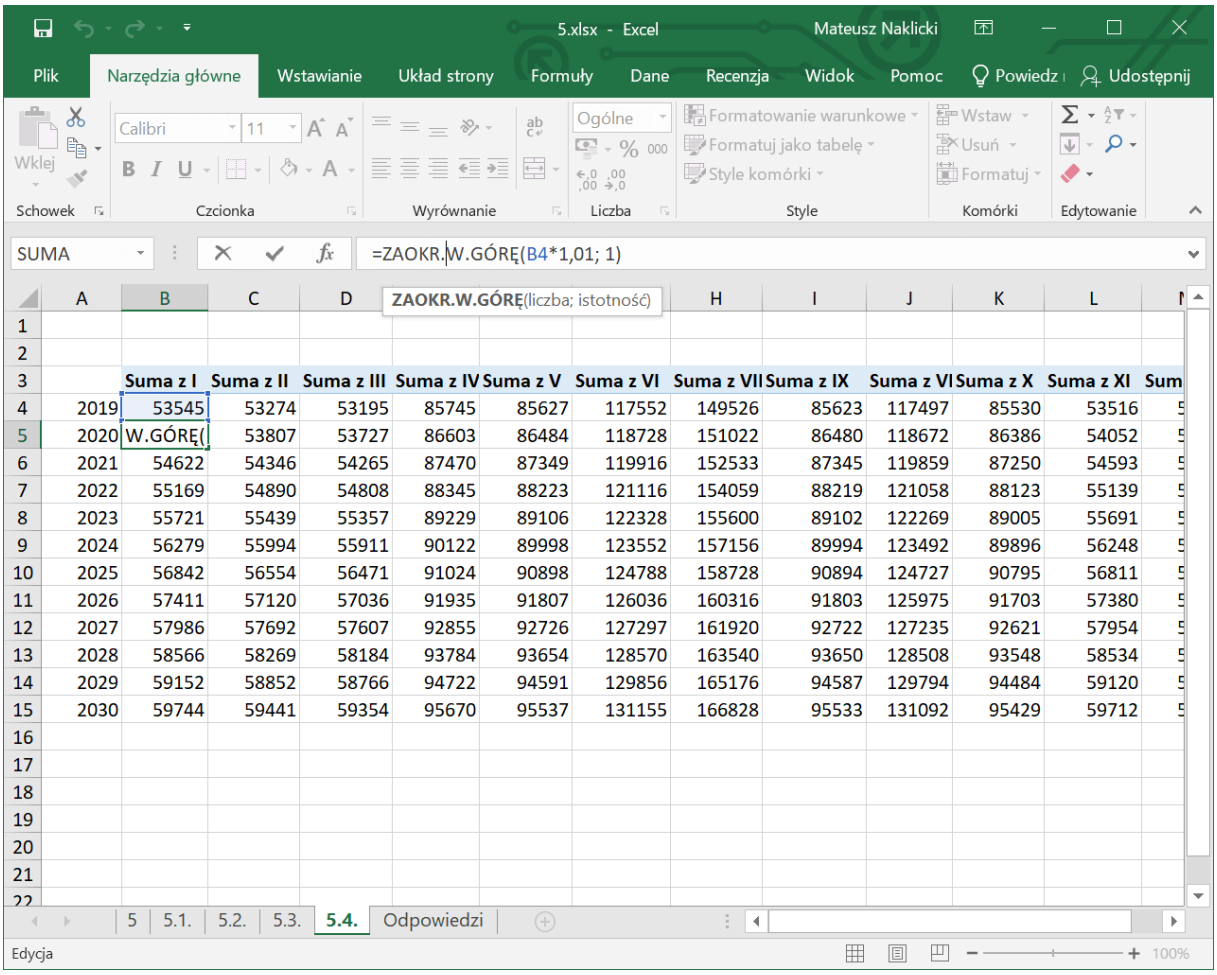
W tym zadaniu także możemy skorzystać z tabeli przestawnej, aby obliczyć zużycie wody każdego miesiąca w 2019 roku. Wystarczy wybrać każdy miesiąc na liście pól, a suma powinna wyliczyć się sama. Będzie to wyglądać w ten sposób:



Wybieramy każdy miesiąc na liście pól w celu uzyskania sumy.

Teraz liczymy symulowane zużycie wody dla następnych lat, które zwiększają się co roku o 1%. Musimy pamiętać o zaokrąglaniu liczby w górę, do czego wykorzystujemy formułę **ZAOKR.W.GÓRĘ**: jako pierwszy argument podajemy liczbę do zaokrąglenia, a jako drugi istotność, czyli czy zaokrąglamy do jedności, dziesiątek czy setek:
=ZAOKR.W.GÓRĘ(B4*1,01; 1).

Robimy to dla każdego roku, aż do 2030:



	A	B	C	D	H	I	J	K	L
3		Suma z I	Suma z II	Suma z III	Suma z VI	Suma z VII	Suma z IX	Suma z VI	Suma z X	Suma z XI
4	2019	53545	53274	53195	117552	149526	85623	117497	85530	53516
5	2020	W.GÓRĘ(I	53807	53727	118728	151022	86480	118672	86386	54052
6	2021	54622	54346	54265	119916	152533	87345	119859	87250	54593
7	2022	55169	54890	54808	121116	154059	88219	121058	88123	55139
8	2023	55721	55439	55357	122328	155600	89102	122269	89005	55691
9	2024	56279	55994	55911	123552	157156	89994	123492	89896	56248
10	2025	56842	56554	56471	124788	158728	90894	124727	90795	56811
11	2026	57411	57120	57036	126036	160316	91803	125975	91703	57380
12	2027	57986	57692	57607	127297	161920	92722	127235	92621	57954
13	2028	58566	58269	58184	128570	163540	93650	128508	93548	58534
14	2029	59152	58852	58766	129856	165176	94587	129794	94484	59120
15	2030	59744	59441	59354	131155	166828	95533	131092	95429	59712

Zaokrąglamy w górę.

Pamiętajmy o sprawdzeniu poprawności naszych wyliczeń: podane zostały w poleceniu poprawne dane dla stycznia 2020 oraz maja 2025, a więc upewnijmy się, czy wszystko dobrze policzyliśmy.

Szukamy teraz miesiąca i roku, w którym jako pierwszy zostanie przekroczona wartość 160 000. Przy takim zadaniu odpowiedź już po chwili jest widoczna: chodzi o **lipiec 2026**.

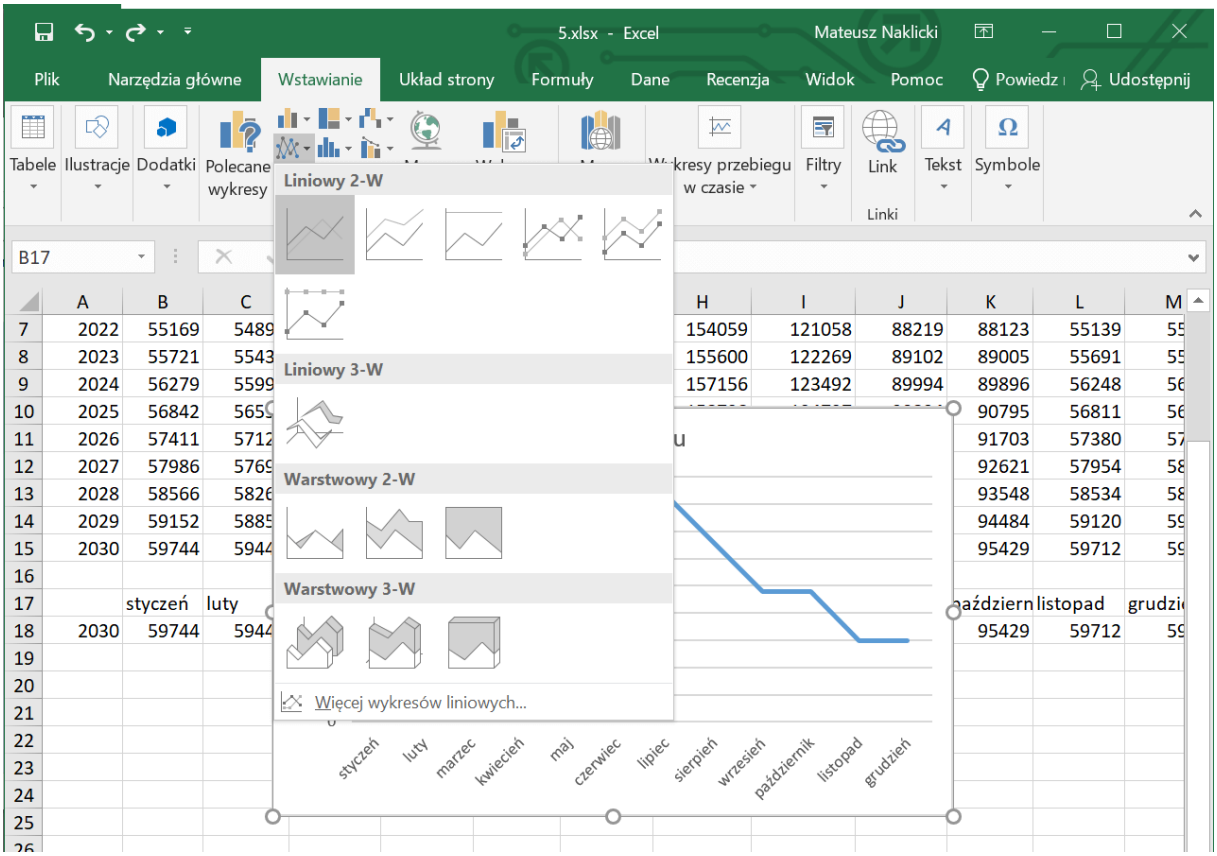
Pamiętajmy jednak, aby zawsze dwa razy sprawdzać dane. Jak może zauważyłeś, z jakiegoś powodu przestawione zostały kolumny Suma z IX i Suma z VIII, przez co mogliśmy popełnić błąd (miesiące nie są zapisane po kolei). Warto to skorygować. W tym wypadku wystarczy odznaczyć i zaznaczyć pola kolejnych miesięcy jeszcze raz na liście pól tabeli przestawnej.

Zostaje nam do stworzenia wykres dla roku 2030. Aby wyglądał schludnie, zapiszmy słownie kolejne miesiące i skopiujmy dane z 2030 roku:

	styczeń	luty	marzec	kwiecień	maj	czerwiec	lipiec	sierpień	wrzesień	październ	listopad	gr
2030	59744	59441	59354	95670	95537	131155	166828	131092	95533	95429	59712	

Kopiujemy dane.

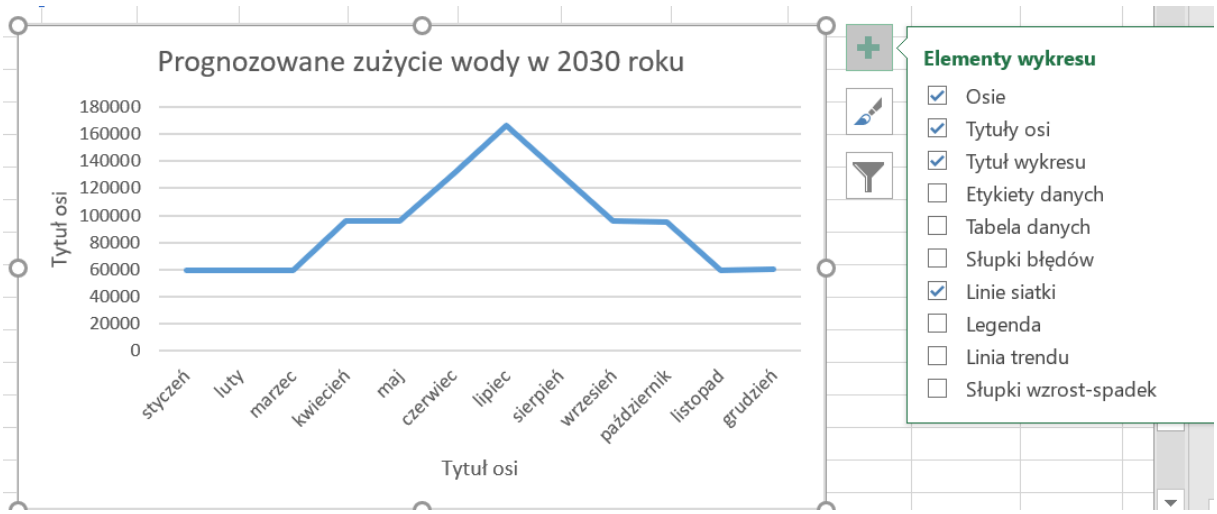
Teraz zaznaczmy te dane i przejdźmy do zakładki Wstawianie -> wykres liniowy:



Tworzymy wykres.

Teraz należy sformatować wykres. Najczęściej na maturze wymaga się tytułu oraz opisów osi. W tym zadaniu nie zostało to sprecyzowane, jednak lepiej to zrobić.

Aby dodać tytuły osi w Excelu 2019, należy kliknąć na wykres, a później na plusa znajdującego się obok:



Dodajemy tytuł osi.

Otrzymujemy finalnie taki wykres:

Ostateczna forma wykresu.

Zadanie 5.5.

Zadanie 5.5. (0–1)

Wodociągi miejskie zaplanowały inwestycję, która począwszy od 2021 roku corocznie w styczniu pozwoli na zwiększanie maksymalnego przepływu o 1000 m^3.

Podaj rok i miesiąc, kiedy pierwszy raz zabraknie wody w mieście po uwzględnieniu tej inwestycji.

Zadanie 5.5 Wodociągi z arkusza maturalnego z informatyki maj 2021

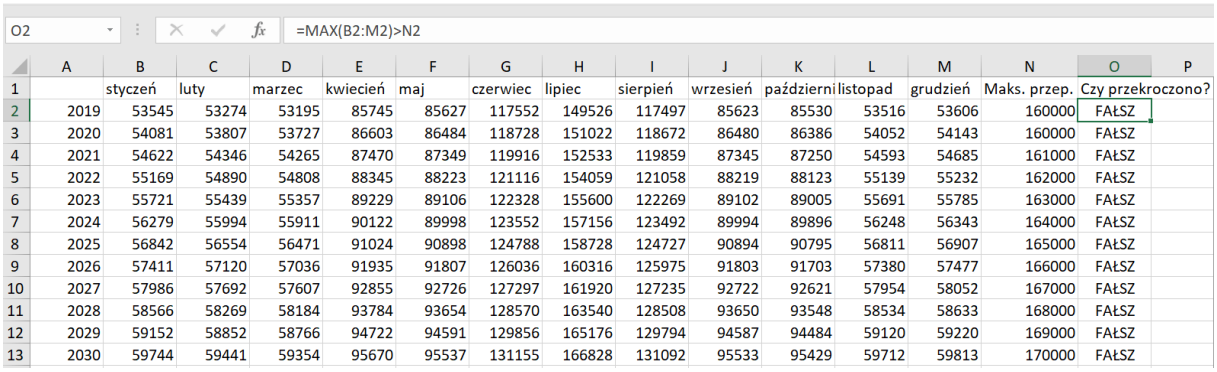
W tym zadaniu wykorzystamy dane przygotowane w poprzednim poleceniu. Skopiujmy dane z lat 2019-2030 do nowego arkusza. W 2019 i 2020 roku maksymalny przepływ wody wyniósł 160 000 m^3, a od 2021 ma zwiększać się każdego roku o 1000 m^3. Zapiszmy to w nowej kolumnie:

	styczeń	luty	marzec	kwiecień	maj	czerwiec	lipiec	sierpień	wrzesień	październi	listopad	grudzień	Maks. przep.
2019	53545	53274	53195	85745	85627	117552	149526	117497	85623	85530	53516	53606	160000
2020	54081	53807	53727	86603	86484	118728	151022	118672	86480	86386	54052	54143	160000
2021	54622	54346	54265	87470	87349	119916	152533	119859	87345	87250	54593	54685	161000
2022	55169	54890	54808	88345	88223	121116	154059	121058	88219	88123	55139	55232	162000
2023	55721	55439	55357	89229	89106	122328	155600	122269	89102	89005	55691	55785	163000
2024	56279	55994	55911	90122	89998	123552	157156	123492	89994	89896	56248	56343	164000
2025	56842	56554	56471	91024	90898	124788	158728	124727	90894	90795	56811	56907	165000
2026	57411	57120	57036	91935	91807	126036	160316	125975	91803	91703	57380	57477	166000
2027	57986	57692	57607	92855	92726	127297	161920	127235	92722	92621	57954	58052	167000
2028	58566	58269	58184	93784	93654	128570	163540	128508	93650	93548	58534	58633	168000
2029	59152	58852	58766	94722	94591	129856	165176	129794	94587	94484	59120	59220	169000
2030	59744	59441	59354	95670	95537	131155	166828	131092	95533	95429	59712	59813	170000

Zapisujemy dane w kolumnach.

Jak widzimy, nigdzie do tej pory nie przekroczyliśmy maksymalnego przepływu. Możemy stworzyć nową kolumnę, w której będziemy sprawdzać, czy w danym roku przekroczony

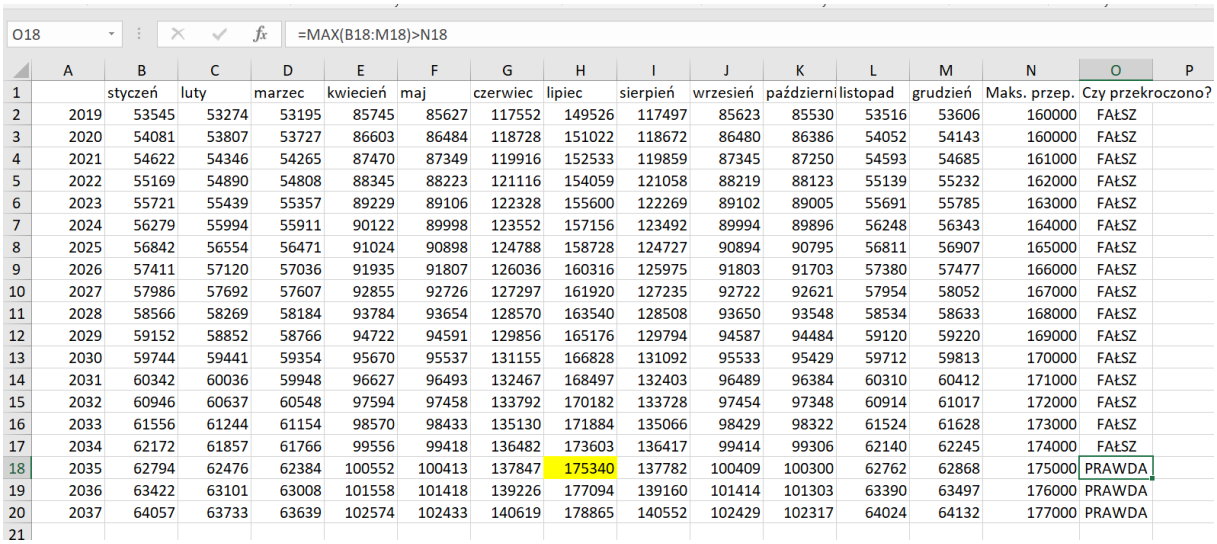
został limit. Będzie ona wyglądać w ten sposób: **=MAX(B2:M2)>N2**, gdzie pod zakresem B2:M2 są wszystkie miesiące, a komórką N2 jest maksymalny przepływ:


O2 =MAX(B2:M2)>N2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		styczeń	luty	marzec	kwiecień	maj	czerwiec	lipiec	sierpień	wrzesień	październi	listopad	grudzień	Maks. przep.	Czy przekroczono?	
2	2019	53545	53274	53195	85745	85627	117552	149526	117497	85623	85530	53516	53606	160000	FAŁSZ	
3	2020	54081	53807	53727	86603	86484	118728	151022	118672	86480	86386	54052	54143	160000	FAŁSZ	
4	2021	54622	54346	54265	87470	87349	119916	152533	119859	87345	87250	54593	54685	161000	FAŁSZ	
5	2022	55169	54890	54808	88345	88223	121116	154059	121058	88219	88123	55139	55232	162000	FAŁSZ	
6	2023	55721	55439	55357	89229	89106	122328	155600	122269	89102	89005	55691	55785	163000	FAŁSZ	
7	2024	56279	55994	55911	90122	89998	123552	157156	123492	89994	89896	56248	56343	164000	FAŁSZ	
8	2025	56842	56554	56471	91024	90898	124788	158728	124727	90894	90795	56811	56907	165000	FAŁSZ	
9	2026	57411	57120	57036	91935	91807	126036	160316	125975	91803	91703	57380	57477	166000	FAŁSZ	
10	2027	57986	57692	57607	92855	92726	127297	161920	127235	92722	92621	57954	58052	167000	FAŁSZ	
11	2028	58566	58269	58184	93784	93654	128570	163540	128508	93650	93548	58534	58633	168000	FAŁSZ	
12	2029	59152	58852	58766	94722	94591	129856	165176	129794	94587	94484	59120	59220	169000	FAŁSZ	
13	2030	59744	59441	59354	95670	95537	131155	166828	131092	95533	95429	59712	59813	170000	FAŁSZ	

Tworzymy nową kolumnę.

Jeśli wrócimy do polecenia okazuje się, że w tym zadaniu nie ma ograniczenia co do lat, tj. rok 2030 nie jest ostatnim, jaki powinniśmy wziąć pod uwagę. Przedłużmy symulację do kolejnych lat:


O18 =MAX(B18:M18)>N18

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		styczeń	luty	marzec	kwiecień	maj	czerwiec	lipiec	sierpień	wrzesień	październi	listopad	grudzień	Maks. przep.	Czy przekroczono?	
2	2019	53545	53274	53195	85745	85627	117552	149526	117497	85623	85530	53516	53606	160000	FAŁSZ	
3	2020	54081	53807	53727	86603	86484	118728	151022	118672	86480	86386	54052	54143	160000	FAŁSZ	
4	2021	54622	54346	54265	87470	87349	119916	152533	119859	87345	87250	54593	54685	161000	FAŁSZ	
5	2022	55169	54890	54808	88345	88223	121116	154059	121058	88219	88123	55139	55232	162000	FAŁSZ	
6	2023	55721	55439	55357	89229	89106	122328	155600	122269	89102	89005	55691	55785	163000	FAŁSZ	
7	2024	56279	55994	55911	90122	89998	123552	157156	123492	89994	89896	56248	56343	164000	FAŁSZ	
8	2025	56842	56554	56471	91024	90898	124788	158728	124727	90894	90795	56811	56907	165000	FAŁSZ	
9	2026	57411	57120	57036	91935	91807	126036	160316	125975	91803	91703	57380	57477	166000	FAŁSZ	
10	2027	57986	57692	57607	92855	92726	127297	161920	127235	92722	92621	57954	58052	167000	FAŁSZ	
11	2028	58566	58269	58184	93784	93654	128570	163540	128508	93650	93548	58534	58633	168000	FAŁSZ	
12	2029	59152	58852	58766	94722	94591	129856	165176	129794	94587	94484	59120	59220	169000	FAŁSZ	
13	2030	59744	59441	59354	95670	95537	131155	166828	131092	95533	95429	59712	59813	170000	FAŁSZ	
14	2031	60342	60036	59948	96627	96493	132467	168497	132403	96489	96384	60310	60412	171000	FAŁSZ	
15	2032	60946	60637	60548	97594	97458	133792	170182	133728	97454	97348	60914	61017	172000	FAŁSZ	
16	2033	61556	61244	61154	98570	98433	135130	171884	135066	98429	98322	61524	61628	173000	FAŁSZ	
17	2034	62172	61857	61766	99556	99418	136482	173603	136417	99414	99306	62140	62245	174000	FAŁSZ	
18	2035	62794	62476	62384	100552	100413	137847	175340	137782	100409	100300	62762	62868	175000	PRAWDA	
19	2036	63422	63101	63008	101558	101418	139226	177094	139160	101414	101303	63390	63497	176000	PRAWDA	
20	2037	64057	63733	63639	102574	102433	140619	178865	140552	102429	102317	64024	64132	177000	PRAWDA	
21																

Przedłużamy symulację do kolejnych lat.

Okazuje się, że wody pierwszy raz zabraknie ponownie w lipcu, w roku 2035. Zapisujemy to jako odpowiedź.

Zadanie 6 – Bitwa

Zadanie 6. Bitwa

W internetowej grze *Bitwa o Kamienną Bramę* bierze udział wielu graczy z całego świata. Każdy gracz może budować różnego rodzaju jednostki (np. robotników, piechurów, łuczników, szpiegów lub magów), które może wykorzystać do budowy wirtualnego księstwa lub wystawić do bitwy. Polem gry jest duża plansza, na której każda jednostka zajmuje pewne miejsce. Każdy gracz może mieć wiele jednostek, np. kilku robotników, drwali, balist i innych. W plikach `gracze.txt`, `klasy.txt` oraz `jednostki.txt` podano aktualny stan wirtualnej planszy. Dane w tych plikach podane są w kolejnych wierszach, w każdym wierszu pola oddzielono znakiem tabulacji. Pierwszy wiersz każdego z plików jest wierszem nagłówkowym.

Plik `gracze.txt` zawiera informację o graczach:

- unikatowy identyfikator będący liczbą całkowitą, numer gracza (*id_gracza*)
- kraj pochodzenia (*kraj*)
- datę dołączenia do gry (*data_dolaczenia*) w formacie rrrr-mm-dd

id_gracza	kraj	data_dolaczenia
1	Japonia	2018-02-15
2	Indie	2017-06-08
3	Stany Zjednoczone	2019-06-10

W pliku `klasy.txt` podano klasy jednostek, jakie gracz może budować. Każda klasa jest opisana przez następujące parametry:

- nazwę klasy jednostek (*nazwa*)
- siłę (*sila*), strzał (*strzal*) oraz magię (*magia*) – trzy atrybuty określające zdolności jednostek tej klasy
- szybkość (*szybkosc*) – prędkość poruszania się jednostek tej klasy

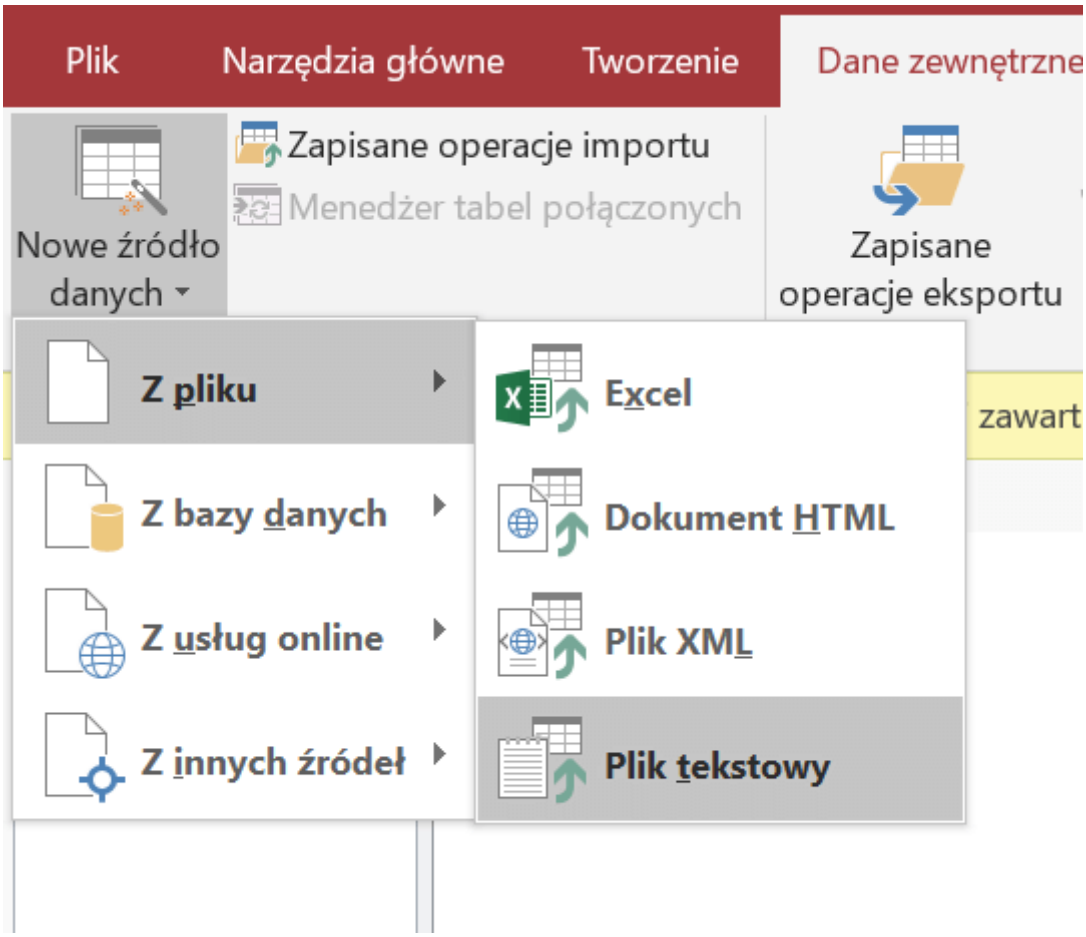
nazwa	sila	strzal	magia	szybkosc
zwiadowca	8	5	0	25
lucznik	5	10	0	12
mag ognia	5	0	15	10
paladyn	20	0	5	20

W pliku `jednostki.txt` podano stan planszy, czyli wszystkie jednostki zbudowane przez graczy. Jeden wiersz pliku opisuje jedną jednostkę za pomocą następujących informacji:

- unikatowy identyfikator będący liczbą naturalną (*id_jednostki*)
- identyfikator gracza, do którego należy jednostka (*id_gracza*)
- nazwę klasy, do której należy jednostka (*nazwa*)
- miejsce jednostki na planszy – jej współrzędne x i y (*lok_x*, *lok_y*)

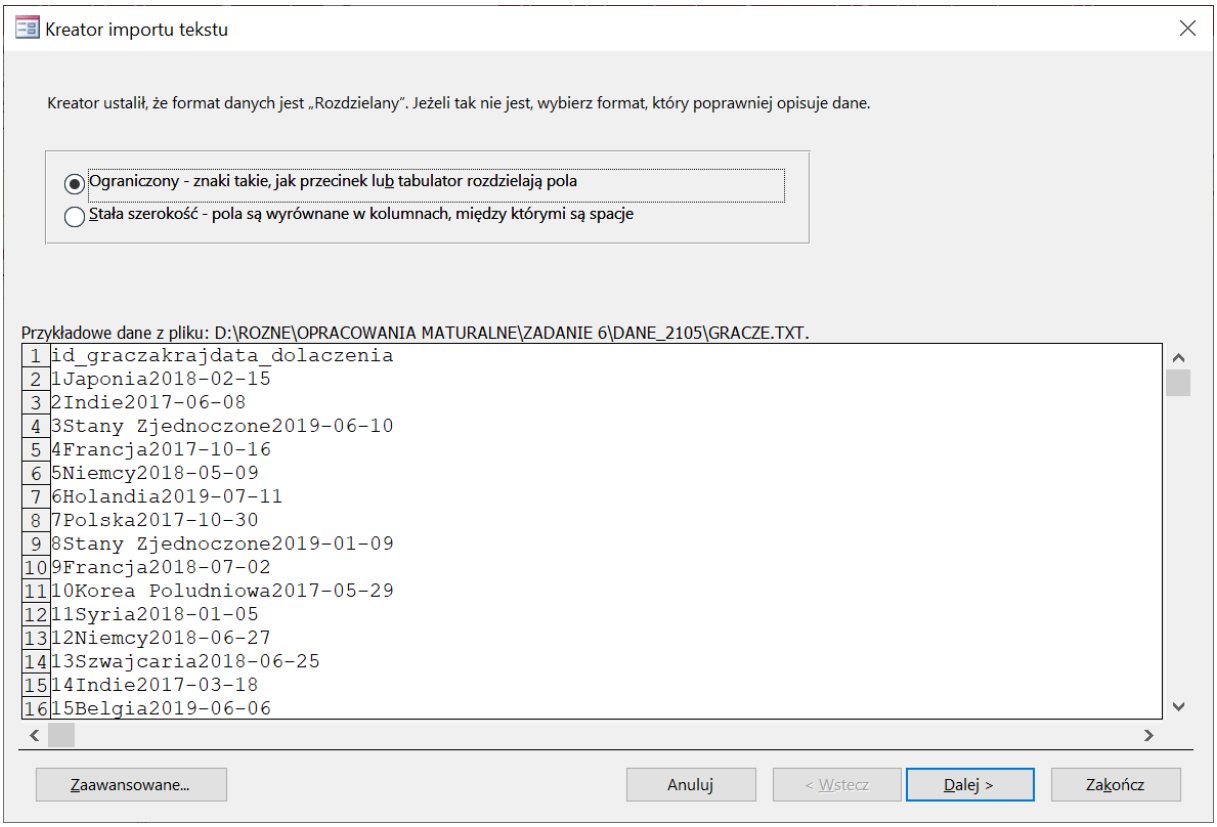
id_jednostki	id_gracza	nazwa	lok_x	lok_y
1	153	piechur	166	30
2	60	topornik	36	44
3	88	drwal	134	88
4	182	kusznik	3	196

Rozpoczynamy od importu danych. Przechodzimy do zakładki Dane zewnętrzne -> Nowe źródło danych -> Z pliku -> Plik tekstowy:



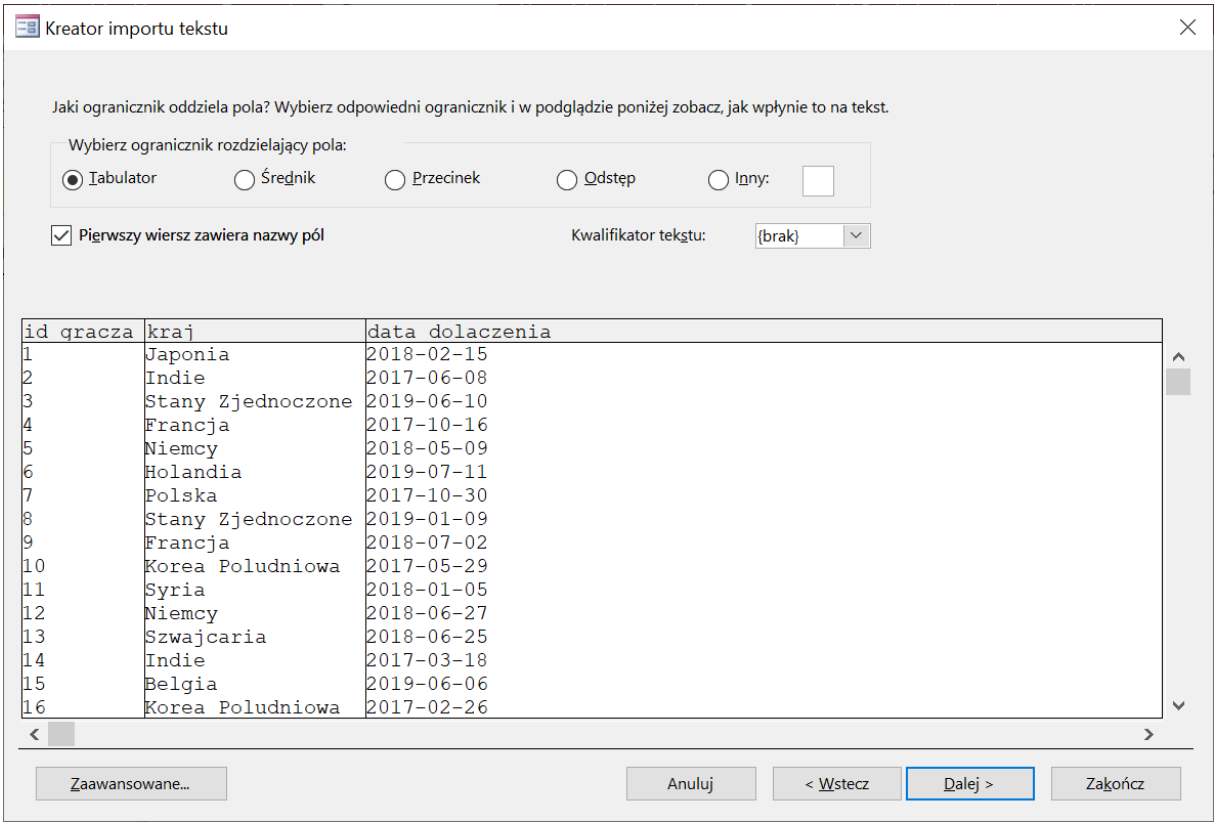
Importujemy dane.

Wybierzmy na początek dołączony plik gracze.txt. W poleceniu zapisane jest, że pola są oddzielone znakiem tabulacji, wybieramy więc format ograniczony:



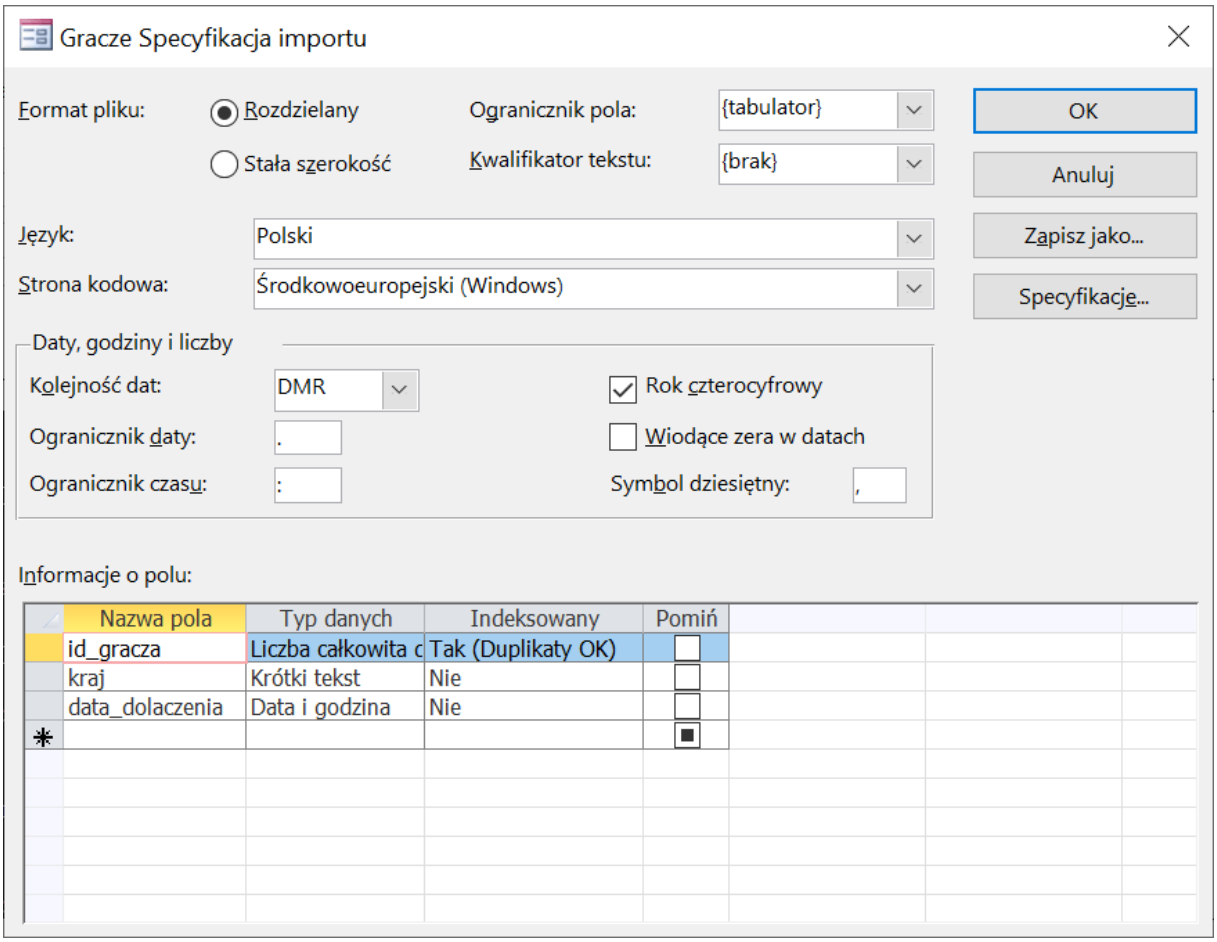
Wybieramy format ograniczony.

Po przejściu dalej wybieramy tabulator jako ogranicznik oraz zaznaczamy pole „Pierwszy wiersz zawiera nazwy pól":



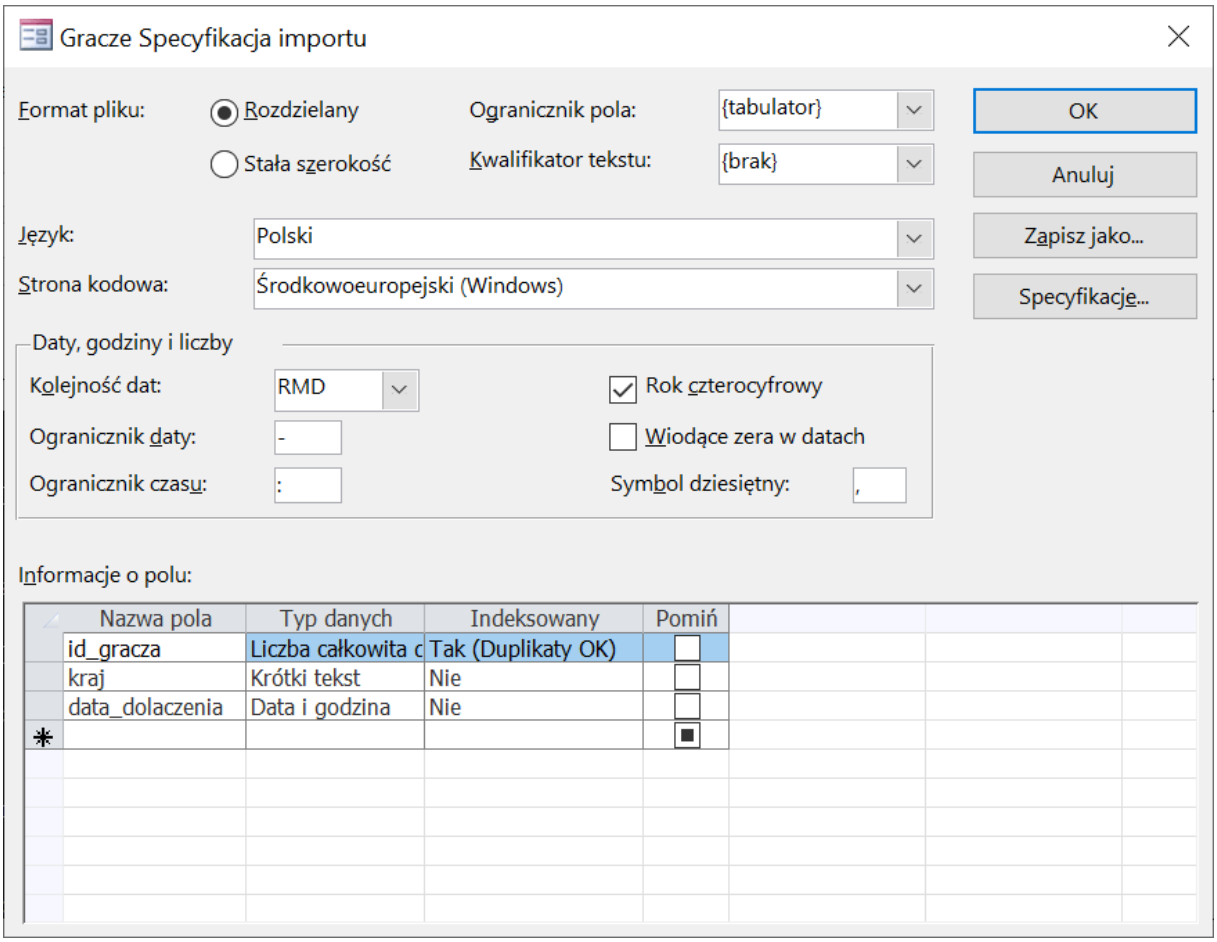
Wybieramy tabulator jako ogranicznik.

W kolejnym oknie sprawdzamy, czy typy danych są prawidłowe dla każdej kolumny. Zazwyczaj Access dobiera właściwe typy, jednak warto się upewnić. Ważną kwestią jest także format dat: aby ustawić odpowiedni, klikamy na Przycisk „Zaawansowane” w lewym dolnym rogu:



Klikamy opcję “Zaawansowane”.

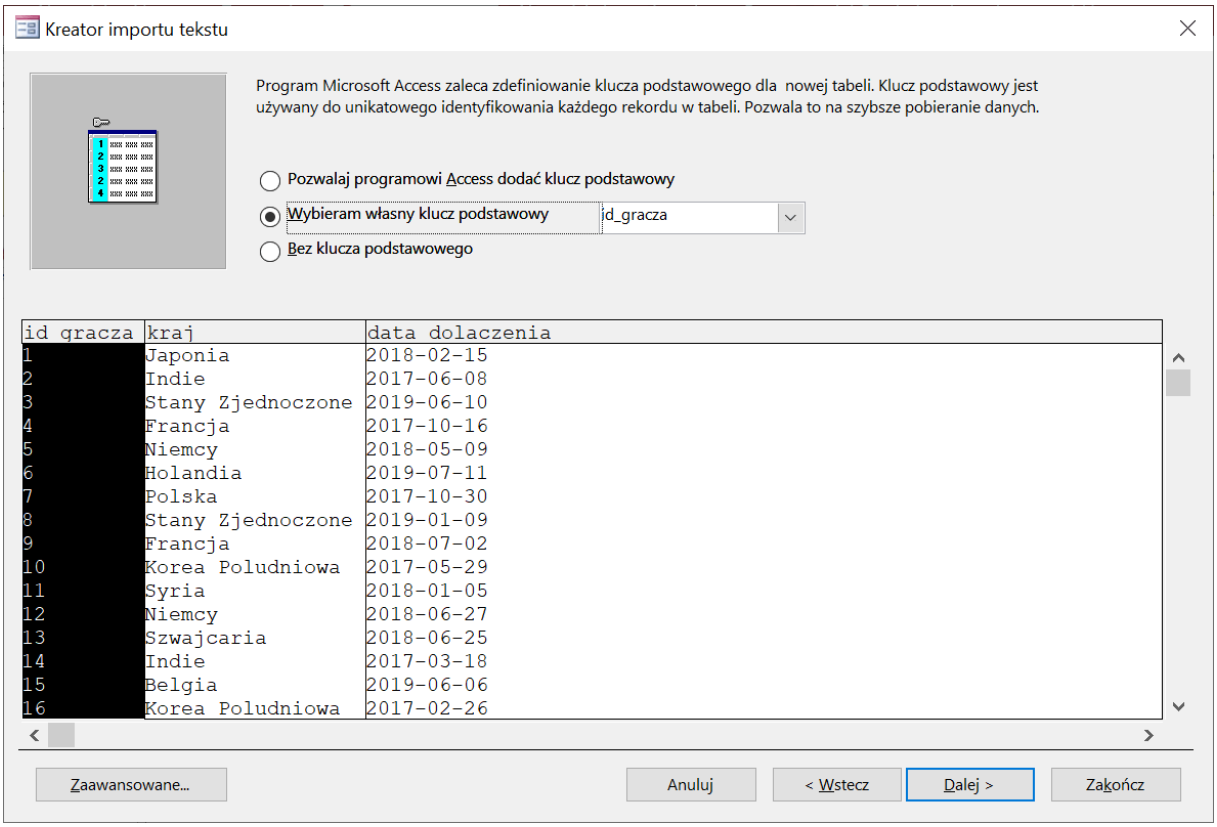
Jak widzimy, domyślnie Access wybrał kolejność dat w formacie dzień, miesiąc, rok, co jest niezgodne z zawartością pliku. Jeśli tego nie edytujemy, otrzymamy później błędy importu. Musimy wybrać rok, miesiąc, dzień, czyli RMD, oraz zapisać ogranicznik daty jako myślnik:



Zapisujemy ogranicznik daty jako myślnik.

Akceptujemy zmiany i przechodzimy dalej.

Pozostaje nam do wybrania klucz podstawowy: jest on dostarczony w kolumnie id_gracza, więc wybierzmy go:



Wybieramy klucz podstawowy.

Przechodzimy dalej i wybieramy nazwę dla tabeli, w naszym przypadku proponowana przez Accessa „Gracze” jest w porządku.

Kroki te powtarzamy dla dwóch kolejnych plików.
W pliku jednostki kluczem podstawowym będzie id_jednostki.
W pliku klasy kluczem podstawowym będzie nazwa jednostki, jako że jest to unikalna wartość dla każdego wpisu.

Po pomyślnym zaimportowaniu wszystkich plików możemy przejść do zadań.

Zadanie 6.1.

Zadanie 6.1. (0–2)

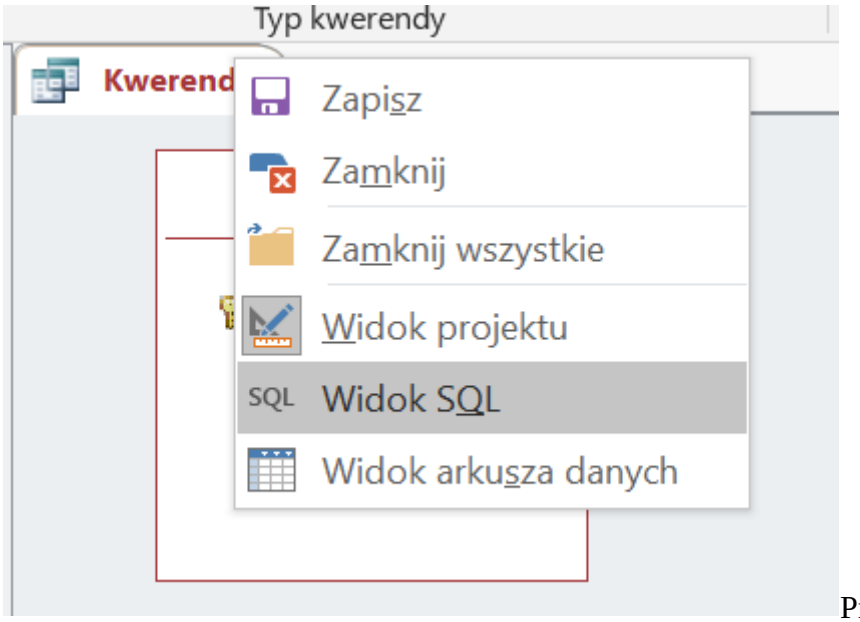
Podaj 5 krajów, z których najwięcej graczy dołączyło do gry w 2018 roku. Dla każdego z tych krajów podaj liczbę graczy, którzy dołączyli w 2018 roku.

Zadanie 6.1 Bitwa z arkusza maturalnego z informatyki maj 2021

Do tworzenia zapytań posłużymy się językiem SQL, który postaramy się jak najlepiej wytłumaczyć. Aby utworzyć kwerendę przechodzimy do zakładki „Tworzenie”, gdzie wybieramy „Projekt kwerendy”. Ukazuje się okno z dostępnymi tabelami. W tym zadaniu potrzebna będzie tabela Gracze, którą dodajemy. Tak powinno wyglądać teraz okno:

Dodajemy tabelę Gracze.

Klikamy prawym przyciskiem na Kwerenda1 i przechodzimy do widoku SQL:



Przechodzimy do widoku SQL.

Tutaj będziemy zapisywać kod potrzebny do wykonania zadań.

Każdą kwerendę wybierającą dane zaczynamy od instrukcji **SELECT**. Następnie podajemy kolumny (jeśli jest więcej niż jedna – oddzielone przecinkiem), z których chcemy uzyskać dane, jak np. kraj czy data_dolaczenia. W tym miejscu można też podać funkcje agregujące (o których za chwilę) lub znak *****, który wybiera wszystkie dostępne kolumny.

Po podaniu tych informacji, zapisujemy źródło danych, które w tym przypadku jest tabelą Gracze. Taki zapis powinien znajdować się już na miejscu, ponieważ wcześniej wybraliśmy tabelę z listy. Chodzi o instrukcję **FROM Gracze**.

Na końcu każdego zapytania znajduje się średnik, jednak nawet gdy o nim zapomnimy, kwerenda wykona się poprawnie.

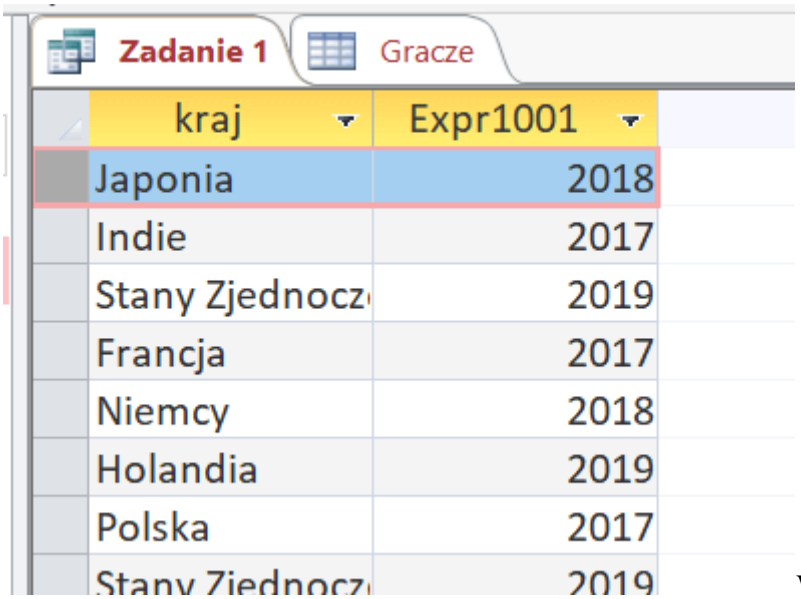
Przykładowa kwerenda będzie wyglądać w ten sposób:

```
1SELECT kraj, data_dolaczenia
2FROM Gracze;
```

W zadaniu jest mowa o graczach, którzy dołączyli w 2018 roku. W języku SQL istnieją funkcje podobne do tych z Excela. Ich listę dla Accessa możemy znaleźć na tej stronie. Aby uzyskać rok, skorzystamy z funkcji YEAR. Aby przetestować działanie funkcji, zapiszmy w kwerendzie zapytanie:

```
1SELECT kraj, YEAR(data_dolaczenia)
2FROM Gracze;
```

Po przejściu do widoku arkusza danych, powinniśmy otrzymać coś takiego:



Widok po przejściu do arkusza danych.

Jak widać, w drugiej kolumnie został wypisany sam rok. Wykorzystamy to później do wybrania osób z 2018 roku.
Nazwa kolumny nie brzmi zachęcająco. Aby to zmienić, po wyrażeniu **YEAR(data_dolaczenia)** możemy zapisać **AS rok**. W ten sposób druga kolumna otrzyma wybraną przez nas nazwę.

Wracając do zadania: należy wybrać osoby z roku 2018. Do wybrania konkretnych wartości służy klauzula **WHERE** wraz z warunkiem, zapisana bezpośrednio po **FROM tabela**. Jeśli interesuje nas tylko rok 2018, zapisujemy to w ten sposób:

```
SELECT kraj, YEAR(data_dolaczenia) AS rok
FROM Gracze
WHERE YEAR(data_dolaczenia) = 2018;
```

Warto zwrócić uwagę na różnice w zapisie między językiem SQL, a klasycznymi językami programowania. Aby sprawdzić, czy wartości są sobie równe, wykorzystujemy pojedynczy znak równości **=.** Natomiast przy sprawdzaniu, czy wartości są różne, zamiast typowego **!=**, w Accessie zapiszemy **<>.**

Teraz należy zliczyć wpisy dla każdego kraju, jako że jeden wpis odpowiada jednemu graczowi. Na szczęście nie musimy robić tego ręcznie, aby uzyskać odpowiedź na zadanie. Służy do tego klauzula **GROUP BY**, dzięki której możemy agregować (pogrupować) dane według np. wartości w danej kolumnie. W naszym przypadku chcemy stworzyć osobny wpis dla każdego kraju, a więc będziemy grupować według kolumny kraj:

```
SELECT kraj, YEAR(data_dolaczenia) AS rok

FROM Gracze

WHERE YEAR(data_dolaczenia) = 2018

GROUP BY kraj;
```

Próba wykonania tej kwerendy zwróci jednak błąd. Gdy grupujemy dane, musimy skorzystać z tzw. funkcji agregujących, ponieważ zbieramy wszystkie wpisy dla danego kraju w jeden wspólny. Główne funkcje agregujące to:

- **COUNT**: zlicza liczbę elementów w grupie
- **SUM**: zlicza sumę elementów w grupie
- **AVG**: zlicza średnią elementów w grupie
- **MAX**: zwraca element z maksymalną wartością
- **MIN**: zwraca element z minimalną wartością

Aby zliczyć graczy z każdego kraju osobno, skorzystamy z funkcji **COUNT**. Zapisujemy je po klauzuli SELECT w taki sposób:

```
SELECT kraj, COUNT(data_dolaczenia) AS liczba_graczy

FROM Gracze

WHERE YEAR(data_dolaczenia) = 2018

GROUP BY kraj;
```

Otrzymaliśmy w ten sposób listę krajów wraz z liczbą graczy z tego kraju. Wystarczy teraz wynik posortować malejąco według liczby graczy i wybrać pięć pierwszych wpisów: służy do tego instrukcja **ORDER BY**, po której podajemy nazwę kolumny, wartość lub numer kolumny. W naszym przypadku chcemy posortować kraje według liczby graczy, dlatego możemy zapisać **ORDER BY** COUNT(data_dolaczenia).
Zapisujemy to na końcu kodu:

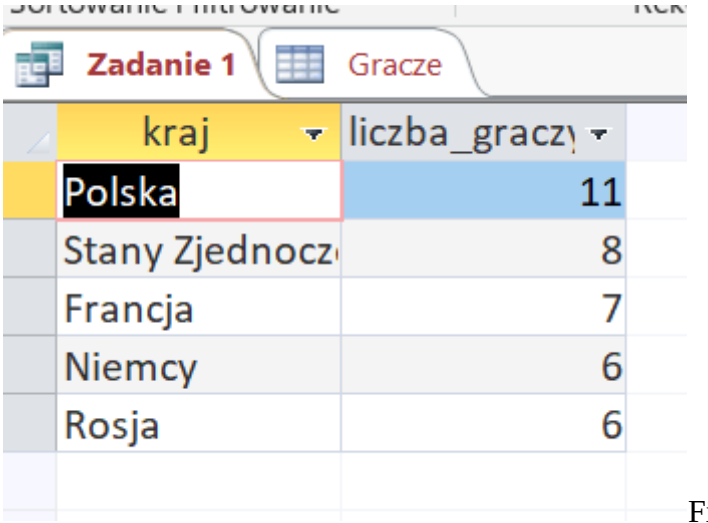
```
SELECT kraj, COUNT(data_dolaczenia) AS liczba_graczy

FROM Gracze

WHERE YEAR(data_dolaczenia) = 2018

GROUP BY kraj

ORDER BY COUNT(data_dolaczenia);
```

Gdy przejdziemy do arkusza danych okazuje się, że domyślnie wpisy posortowane są malejąco. Aby to zmienić, po **ORDER BY instrukcja** zapisujemy **DESC** (od ang. descending, malejąco): ORDER BY COUNT(data_dolaczenia) DESC;

Otrzymujemy wpisy krajów z największą liczbą graczy. Możemy skopiować już pięć pierwszych do pliku wyniki6.txt. Jeśli chcemy być bardzo dokładni, istnieje instrukcja, która pozwala ograniczyć liczbę wypisanych wierszy. W Accessie jest to TOP, po której zapisujemy ile wpisów chcemy uzyskać. Zapisujemy ją bezpośrednio po SELECT:

```
SELECT TOP 5 kraj, COUNT(data_dolaczenia) AS liczba_graczy

FROM Gracze

WHERE YEAR(data_dolaczenia) = 2018

GROUP BY kraj

ORDER BY COUNT(data_dolaczenia) DESC;
```

Taką odpowiedź powinniśmy otrzymać:



Finalna forma tabeli.

Zadanie 6.2.

Zadanie 6.2. (0–2)

Podaj sumę wartości pola strzał (*strzal*) dla każdej klasy jednostek, które mają „elf” jako część nazwy.

Zadanie 6.2 Bitwa z arkusza maturalnego z informatyki maj 2021

Tworzymy nowy projekt kwerendy, tym razem dla tabeli klasy. Polecenie wskazuje na to, że ponownie musimy skorzystać z funkcji agregującej, w tym wypadku będzie to **SUM**.
Zsumujemy kolumnę **strzal** dla każdej klasy posiadającej w nazwie słowo **elf**.
Aby znaleźć takie klasy, wykorzystamy klauzulę **LIKE** jako warunek. Pozwala ona na porównywanie napisów z wykorzystaniem tzw. symboli wieloznacznych. W Accessie najważniejszymi symbolami są:

- znak *****: oznaczający dowolną liczbę dowolnych znaków
- znak **?**: oznaczający dowolny pojedynczy znak

Aby znaleźć nazwy, które zawierają w sobie słowo elf, zapiszemy **LIKE** “***elf***“, co oznacza, że zarówno przed, jak i po napisie elf może wystąpić dowolna liczba znaków.

Finalnie kwerenda wygląda w ten sposób:

```
SELECT SUM(strzal) as Suma

FROM Klasy

WHERE nazwa LIKE "*elf*";
```

Po jej wykonaniu otrzymujemy odpowiedź 35.

Zadanie 6.3.

Zadanie 6.3. (0–2)

Podaj numery graczy, którzy **nie mają** artylerzystów (jednostek o nazwie *artylerzysta*). Numery podaj w porządku rosnącym.

Zadanie 6.3 Bitwa z arkusza maturalnego z informatyki maj 2021

W tym zadaniu wykorzystamy dwie kwerendy. W pierwszej z nich znajdziemy numery graczy, którzy posiadają artylerzystów, a dopiero w drugiej wyszukamy tych, którzy takich jednostek nie posiadają.

Stwórzmy kwerendę, która korzystać będzie z tabeli jednostki. Aby wybrać graczy, którzy mają na stanie artylerzystów, wykorzystamy klauzulę WHERE:

```
SELECT id_gracza

FROM Jednostki

WHERE nazwa = "artylerzysta";
```

W ten sposób uzyskaliśmy identyfikatory wszystkich graczy z jednostkami artylerzysty.

Aby wydobyć graczy bez jednostek artylerzysty, wykorzystamy powyższa kwerendę. Najpierw musimy wydobyć identyfikator każdego z graczy, a więc skorzystamy z tabeli Gracze:

```
SELECT id_gracza
FROM Gracze;
```

Teraz jako warunek wybierzemy brak identyfikatora gracza w kwerendzie, którą zapisaliśmy powyżej:

```
SELECT id_gracza
FROM Gracze
WHERE id_gracza NOT IN

(SELECT id_gracza

FROM Jednostki

WHERE nazwa = "artylerzysta");

```

W taki sposób połączyliśmy dwie kwerendy i otrzymaliśmy odpowiedź:

Zadanie-3

id_gracza
22
24
29
35
36
37
38
47
54
61
64
72
110
114
115
122
123
138
141
167
*

Odpowiedź po połączeniu dwóch kwerend.

Zadanie 6.4.

Zadanie 6.4. (0–2)

Jeden krok jednostki to przejście o 1 w dowolnym z czterech kierunków (północ, południe, wschód lub zachód). W jednej turze jednostka może wykonać co najwyżej tyle kroków, ile wynosi jej *szybkosc*. Innymi słowy jednostka w ciągu jednej tury może przemieścić się z punktu (x,y) do punktu (x_1,y_1), jeśli $|x - x_1| + |y - y_1| \leq szybkosc$.

Tytułowa *Kamienna Brama* znajduje się w miejscu (100,100). Wyszukaj jednostki, które mogą w jednej turze dojść do Bramy, i podziel je na poszczególne klasy. Utwórz zestawienie, które dla każdej klasy poda jej nazwę oraz liczbę jednostek z tej klasy mogących w jednej turze osiągnąć Bramę.

Zadanie 6.4 Bitwa z arkusza maturalnego z informatyki maj 2021

Skupmy się na wyborze jednostek, które mogą dojść do Kamiennej Bramy w jednej turze. Najważniejszy dla nas jest podany warunek: |x – x1| + |y – y1| ≤ szybkosc.
Kamienna Brama znajduje się w lokacji o koordynatach (100, 100), a jednostka w (lok_x, lok_y). Warunek będzie wyglądać w ten sposób: |100 – lok_x| + |100 – lok_y| ≤ szybkosc.
Teraz stworzenie odpowiedniej kwerendy jest już formalnością. Pamiętajmy jedynie, że wartość bezwzględna w Accessie jest oznaczana pod postacią funkcji **abs**.

Przy tworzeniu kwerendy potrzebować będziemy lokalizację, klasę i szybkość jednostek. Tworząc kwerendę wybierzemy więc tabele Jednostki oraz Klasy. Dzięki temu w kodzie zapisane zostaje złączenie dwóch tabel w formie **INNER JOIN**.

Kwerenda, która wybierze nam wartości dotyczące jednostek będzie wyglądać tak:

```
1SELECT id_jednostki, lok_x, lok_y, szybkosc
2
3FROM Jednostki  INNER JOIN Klasy  ON Jednostki.nazwa = Klasy.nazwa
4
5WHERE abs(100-lok_x) + abs(100-lok_y) &amp;lt;= szybkosc;
```

Możemy sprawdzić poprawność danych, przechodząc do arkusza:

Jednostki | Zadanie-4

id_jednostki	lok_x	lok_y	szybkosc
139	102	107	10
422	107	98	25
540	96	104	12
825	105	101	8
838	102	107	25
848	106	111	25
901	102	85	25
987	99	103	25
1325	110	88	25

Sprawdzamy poprawność danych przechodząc do arkusza.

Jak widzimy, warunek wybiera odpowiednie dane. Wróćmy do treści zadania.

Musimy teraz pogrupować jednostki na klasy, a następnie zliczyć jednostki, które są w stanie dojść do lokalizacji (100, 100). Zróbmy to:

```
SELECT Jednostki.nazwa, COUNT(*) As Liczba_jednostek

FROM Jednostki  INNER JOIN Klasy  ON Jednostki.nazwa = Klasy.nazwa

WHERE abs(100-lok_x) + abs(100-lok_y) &amp;lt;= szybkosc

GROUP BY Jednostki.nazwa;
```

Otrzymujemy odpowiedź:

Jednostki | Zadanie-4

nazwa	Liczba_jednostek
architekt	1
artylerzysta	4
balista	2
ciemny elf	1
drwal	3
elfi czarodziej	1
goniec	2
ifryt	1
kaplan	2
kawalerzysta	7
konny lucznik	4
kusznik	1
lekki jezdziec	19
lucznik	1
mag powietrza	3
mag wody	2
paladyn	1
piechur	7
pikinier	5
robotnik	5
topornik	5
wysoki elf-gwa	1
zwiadowca	4

Rekord: 1 z 23 Bez filtru Wys

Ostateczna odpowiedź.

Zadanie 6.5.

Zadanie 6.5. (0–4)

Jeśli w pewnej lokalizacji znajdują się jednostki więcej niż jednego gracza, toczy się tam (jedna) bitwa. Oblicz:

a) ile bitew ma miejsce na planszy,

b) w ilu bitwach biorą udział gracze z Polski.

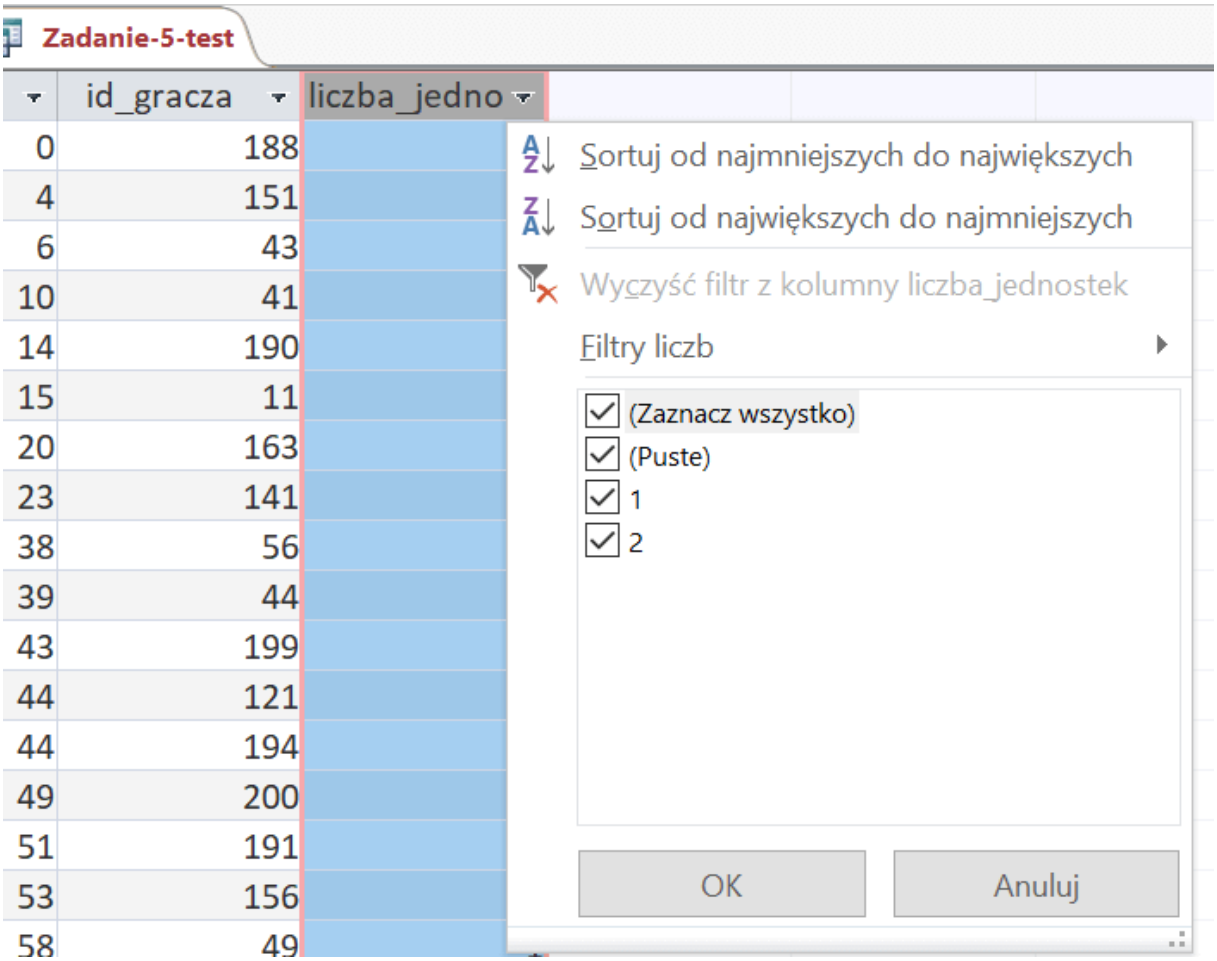
Uwaga: zauważ, że w jednej lokalizacji może się znajdować więcej niż jedna jednostka tego samego gracza.

Zadanie 6.5 Bitwa z arkusza maturalnego z informatyki maj 2021

W pierwszym podpunkcie wystarczyłoby pogrupować jednostki według lokalizacji, a następnie je zliczyć i wybrać takie, których zliczenie zwróciło 2 lub więcej (znajduje się w tym miejscu więcej niż jedna jednostka). Co jednak, jeśli wszystkie jednostki na polu będą należeć do tego samego gracza? Sprawdźmy, czy takie sytuacje mają w ogóle miejsce:

```
1SELECT lok_x, lok_y, id_gracza, COUNT(*) AS liczba_jednostek
1FROM Jednostki
2
3GROUP BY lok_x, lok_y, id_gracza;
```

Przechodząc do arkusza danych, kliknijmy na trójkąt obok kolumny liczba_jednostek. Powinniśmy ujrzeć coś takiego:



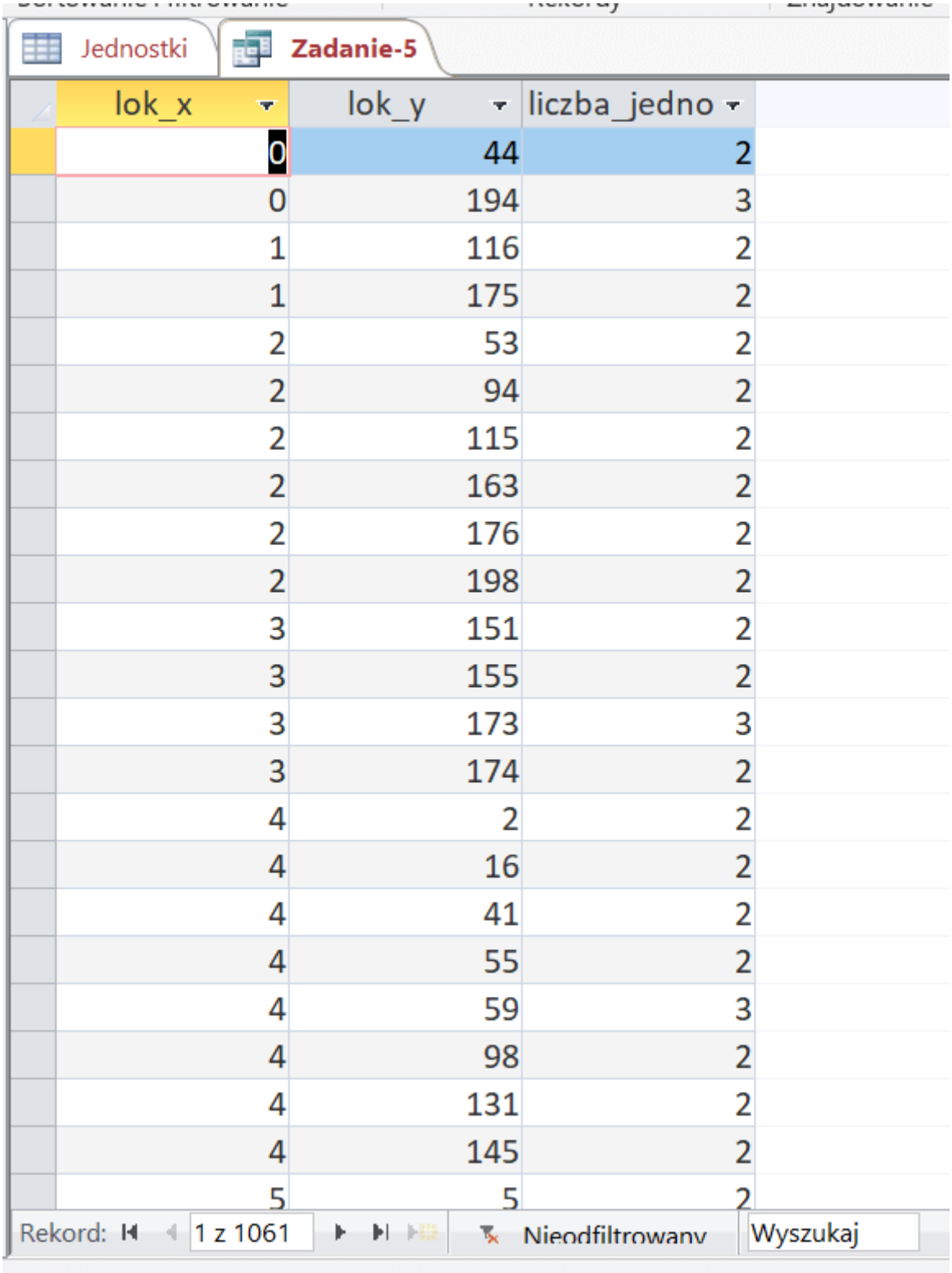
Klikamy na trójkąt obok kolumny liczba_jednostek.

Odznaczmy wartość 1. Otrzymaliśmy listę pozycji, w których dwie jednostki należą do tego samego gracza. Na naszej liście bitew musimy wziąć to pod uwagę, stwórzmy w tym celu tabelę wykluczającą więcej niż jedną jednostkę tego samego gracza. Możemy w tym celu edytować powyższą kwerendę, dodając na końcu klauzulę **HAVING**, która działa podobnie jak **WHERE**, tyle że jest przystosowana do działaniach na funkcjach agregujących. Znaczy to tyle, że po **HAVING** możemy zapisać warunek z **COUNT(*)**, czego nie można zrobić przy **WHERE**:

```
1SELECT lok_x, lok_y, id_gracza, COUNT(*) AS liczba_jednostek
2
3FROM Jednostki
4
5GROUP BY lok_x, lok_y, id_gracza
6
7HAVING COUNT(*) = 1;
```

Otrzymujemy tu tabelę, z której teraz powinniśmy zliczyć mające miejsce bitwy. Zróbmy to w nowej kwerendzie, przy której dane wybierzemy z wcześniej stworzonej:

```
1SELECT lok_x, lok_y, COUNT(*) As liczba_jednostek
2
3FROM [Zadanie-5-jednostki]
4
5GROUP BY lok_x, lok_y
6
7HAVING COUNT(*) &amp;gt; 1;
```



Wybieramy dane ze starej kwerendy.

Już widzimy, ile bitew trwa na mapie. 1061 – widać to w dolnej części Accessa. Zapisujemy to jako odpowiedź i przechodzimy do kolejnego podpunktu.

Tutaj sprawa się komplikuje. Bitwy, które nas interesują, są toczone przez Polaków. Oznacza to, że do kwerendy u mnie nazwanej Zadanie-5-jednostki, należy dołączyć tabelę Gracze i sprawdzić kraj uczestnika. Trzeba jednak też sprawdzić, czy występują takie bitwy, w których

Polacy walczą ze sobą tj. w bitwie uczestniczy więcej niż jedna osoby z Polski. Takie przypadki musimy liczyć jako jeden, ponieważ interesuje nas liczba bitw, w których uczestniczą Polacy, a nie liczba osób z Polski uczestnicząca w bitwach.
Aby uzyskać liczbę bitew, w których uczestniczy więcej niż jedna osoba z Polski, stwórzmy nową kwerendę:

```
SELECT lok_x, lok_y, kraj, COUNT(*) AS liczba_jednostek

FROM [Zadanie-5-jednostki] INNER JOIN Gracze ON [Zadanie-5-jednostki].id_gracza = Gracze.id_gracza

WHERE kraj LIKE "Polska"

GROUP BY lok_x, lok_y, kraj

HAVING COUNT(*) > 1;
```

Powinniśmy otrzymać 19 wpisów:

Jednostki | Zadanie-5-jednostki | **Zadanie-5-kraje**

lok_x	lok_y	kraj	liczba_jedno
7	126	Polska	2
27	166	Polska	2
49	132	Polska	2
50	199	Polska	2
56	186	Polska	2
58	192	Polska	2
61	109	Polska	2
77	82	Polska	2
90	199	Polska	2
93	66	Polska	2
111	19	Polska	2
141	125	Polska	2
150	26	Polska	2
153	146	Polska	2
167	194	Polska	2
168	57	Polska	2
169	173	Polska	2
198	115	Polska	2
199	92	Polska	2

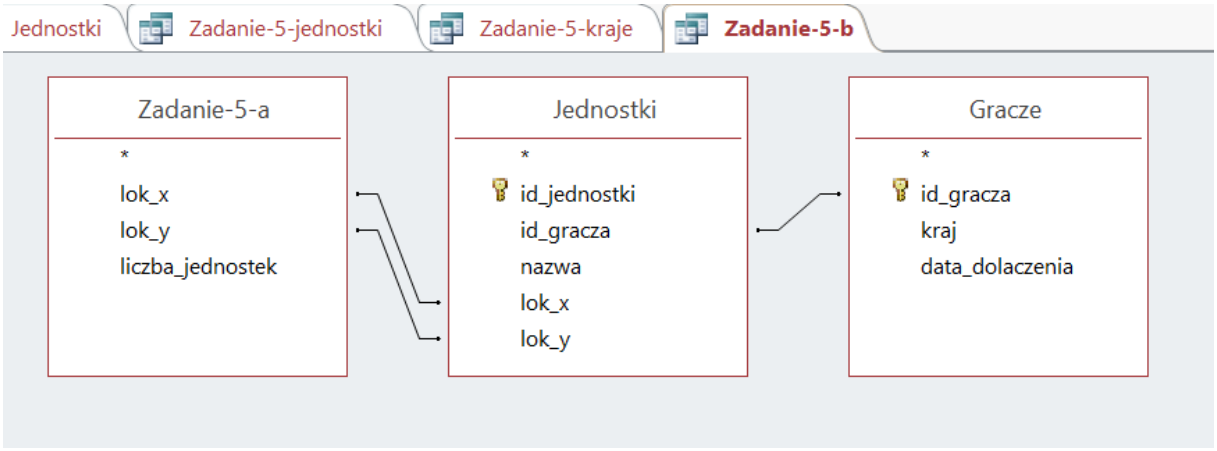
Rekord: 1 z 19 | Filtrowane | Wyszukaj

Otrzymujemy 19 wpisów.

Później odliczymy od wyniku 19, aby wykluczyć właśnie te bratobójcze bitwy.

Teraz zajmiemy się wyznaczeniem liczby bitew toczonych przez Polaków.

W tym celu wykorzystamy dane odnośnie toczących się bitew z podpunktu a, tabele Jednostki oraz Gracze. Połączmy je graficznie tak jak na zrzucie:



Łączymy tabele Jednostki oraz Gracze.

Uzyskaliśmy w ten sposób informacje odnośnie każdej bitwy, tj. kto ją toczy i z jakiego kraju pochodzi.

Przejdźmy do kodu i zapiszmy zliczanie rekordów za pomocą **COUNT(*)** oraz warunek **WHERE kraj LIKE “Polska”**:

```
SELECT COUNT(*) as bitwy_polakow
FROM (Jednostki INNER JOIN Gracze ON Jednostki.id_gracza = Gracze.id_gracza)
INNER JOIN [Zadanie-5-a] ON (Jednostki.lok_y = [Zadanie-5-a].lok_y) AND
(Jednostki.lok_x = [Zadanie-5-a].lok_x)
WHERE kraj LIKE "Polska";
```

Wyciągnęliśmy tu liczbę Polaków, którzy toczą bitwę. Wynosi ona 264. Są tutaj jednak też uwzględnione wspomniane wcześniej „bratobójcze” walki między Polakami, które powinniśmy liczyć pojedynczo. Dlatego w wyniku zapisujemy 245, ponieważ odejmujemy wyznaczone 19 bitw, w których osoby z Polski toczą walkę między sobą (muszą być one liczone pojedynczo).