

## INFORMATYKA klasa 4 LO

poziom rozszerzony

Plan wynikowy z wymaganiami edukacyjnymi przedmiotu informatyka dla klasy IV liceum ogólnokształcącego i technikum w zakresie rozszerzonym, uwzględniający kształcone umiejętności i treści podstawy programowej

Temat	Ocena dopuszczająca. Uczeń:	Ocena dostateczna Uczeń:	Ocena dobra Uczeń:	Ocena bardzo dobra Uczeń:	Ocena celująca Uczeń:
<b>I. Algorytmika i programowanie</b>					
Więcej o liczbach, czyli logarytm, sito Eratostenesa i schemat Hornera.	<ul style="list-style-type: none"> <li>– zna definicję logarytmu</li> <li>– zna ogólną ideę algorytmu sita Eratostenesa (wykreślanie wielokrotności kolejnych liczb pierwszych)</li> <li>– potrafi zapisać naiwny (“pierwiastkowy”) algorytm sprawdzania pierwszości</li> </ul>	<ul style="list-style-type: none"> <li>– zna przykłady sytuacji, w których używa się logarytmu</li> <li>– potrafi prześledzić działanie algorytmu sita Eratostenesa na konkretnych danych</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi rozpoznać złożoność logarytmiczną w algorytmach</li> <li>– potrafi zapisać w pseudojęzyku algorytm sita Eratostenesa</li> <li>– potrafi zapisać w pseudojęzyku schemat Hornera</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zaimplementować sito Eratostenesa i schemat Hornera w wybranym języku programowania</li> <li>– zna złożoność algorytmu sita Eratostenesa i porównuje ją z “pierwiastkowym” algorytmem</li> </ul>	
Rekursja raz jeszcze, czyli dziel i zwyciężaj.	<ul style="list-style-type: none"> <li>– potrafi zasymulować (prześledzić działanie) prostej funkcji rekurencyjnej</li> </ul>	<ul style="list-style-type: none"> <li>– zna warunki działania rekursji, potrafi przekształcać proste algorytmy (z jednym wywołaniem rekurencyjnym) z iteracyjnych na rekurencyjne i odwrotnie</li> </ul>	<ul style="list-style-type: none"> <li>– prawidłowo analizuje algorytmy rekurencyjne typu „dziel i zwyciężaj”, zna i umie stosować warunki działania rekursji w projektowaniu algorytmów</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi samodzielnie zaprojektować algorytm opierający się na rekursji z dwoma wywołaniami typu „dziel i zwyciężaj”</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zanalizować nietypowe wywołania rekurencyjne (np. na więcej niż dwóch podzadaniach albo na różniących się od siebie podzadaniach)</li> </ul>
Sortowanie przez scalanie, czyli pierwszy efektywny algorytm sortowania.	<ul style="list-style-type: none"> <li>– zna ogólną ideę algorytmu sortowania przez scalanie (podzielić na dwie połowy, rekursja, scalenie)</li> </ul>	<ul style="list-style-type: none"> <li>– zna ogólną ideę scalania, potrafi podać złożoność algorytmu (<math>n \log n</math>)</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zapisać cały algorytm sortowania przez scalanie w pseudojęzyku</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zapisać algorytm sortowania przez scalanie w wybranym języku programowania bez błędów i pilnując przypadków brzegowych</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi uargumentować złożoność algorytmu sortowania przez scalanie</li> </ul>
Sortowanie szybkie, czyli jak się sortuje w praktyce.	<ul style="list-style-type: none"> <li>– zna ogólną ideę algorytmu szybkiego sortowania (przestawić</li> </ul>	<ul style="list-style-type: none"> <li>– wie, czym jest klucz i zna ideę operacji przestawiania, zna</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zapisać cały algorytm sortowania przez scalanie w pseudojęzyku</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi zapisać algorytm sortowania przez scalanie w wybranym języku</li> </ul>	<ul style="list-style-type: none"> <li>– potrafi przedyskutować i uargumentować złożoność algorytmu przy różnych wyborach klucza</li> </ul>

	elementy, a potem wywołanie rekurencyjne)	złożoność algorytmu w średnim wypadku		programowania, rozumie korzyść z losowania klucza	(najlepszym/najgorszy m)
Specjalne elementy w tablicy, czyli największy, najmniejszy i lider.	- potrafi znaleźć oddzielnie element najmniejszy i największy - wie, czym jest lider i dominanta, potrafi podać przynajmniej jeden algorytm znajdowania dominanty w zbiorze	– potrafi prześledzić działanie (przeanalizować) algorytm jednoczesnego znajdowania najmniejszego i największego elementu oraz algorytm znajdowania lidera	– potrafi zapisać w pseudojęzyku algorytm jednoczesnego znajdowania najmniejszego i największego elementu oraz algorytm znajdowania lidera i podać ich złożoność	– potrafi uargumentować liczbę porównań w algorytmie jednoczesnego znajdowania największego/najmniejszego elementu oraz poprawność algorytmu znajdowania lidera	
Tablice dynamiczne i listy wiązane, czyli kiedy tablica nie wystarcza	– zna ogólną ideę tablicy dynamicznej (tablica + zmienny rozmiar) oraz listy wiązanej	– opisuje operacje dostępne na tablicy dynamicznej i liście wiązanej, potrafi przedstawić na rysunku budowę i działanie listy wiązanej	– potrafi przeanalizować prosty algorytm, w którym występują operacje na tablicach dynamicznych / listach wiązanych, opisuje rozwiązanie problemu Flawiusza za pomocą listy wiązanej	– swobodnie analizuje i projektuje algorytmy zawierające tablice dynamiczne i listy wiązane	– stosuje tablice dynamiczne i listy wiązane (np. <code>std::vector</code> , <code>std::list</code> , listy w Pythonie) w swoich programach
Stos i kolejka, czyli struktury proste i bardzo użyteczne.	– zna ogólną ideę stosu i kolejki	– opisuje operacje na stosie i kolejce	– analizuje proste algorytmy, w których występują struktury stosu i kolejki, aplikuje strukturę stosu do rozwiązania problemu ONP	– swobodnie projektuje analizuje algorytmy, w których występują struktury stosu i kolejki	– posługuje się typem stos/kolejka w wybranym języku programowania w swoich programach
Grafy, czyli co mają wspólnego sieć społecznościowa i paryskie metro	– potrafi rozpoznać graf w sytuacjach praktycznych	– podaje przykłady konkretnych sytuacji modelowanych za pomocą grafu, zna definicję grafu	– zna implementację grafu przez macierze i listy sąsiedztwa	- potrafi analizować i projektować algorytmy, w których występują grafy - potrafi porównać implementację macierzową i listową, podając ich zalety i wady	– potrafi samodzielnie zaimplementować graf w wybranym języku programowania

Wyszukiwanie idola, czyli o tym, jak spóźniliśmy się na turniej tenisowy	– wie, czym jest element-idol, potrafi znaleźć element-idol dowolnym algorytmem (np. pytając o wszystkie pary)	– zna ogólną ideę algorytmu wyszukiwania idola („pytaj jeden element do momentu przegranej”)	– potrafi zapisać w pseudojęzyku algorytm wyszukiwania idola	– potrafi uargumentować złożoność algorytmu wyszukiwania idola	
Najkrótsze ścieżki w grafie, czyli jak komputery znajdują drogę	– potrafi podać, czym jest ścieżka w grafie i pokazać jej długość na przykładzie drogi	– zna ogólną ideę algorytmu wyszukiwania ścieżki („najpierw elementy o odległości 1, potem 2, i tak dalej”)	– potrafi zapisać w pseudojęzyku szkic algorytmu BFS	– potrafi zapisać w pseudojęzyku pełny algorytm BFS z zastosowaniem kolejki i prosto uzasadnić jego złożoność	– implementuje działający algorytm BFS w wybranym języku programowania
Programowanie strukturalne kontra obiektowe, czyli dane w centrum uwagi	– rozumie ideę programowania obiektowego („funkcje przypisane do danych”)	– rozumie pojęcie klasy/obiektu oraz pola i metody	– wyjaśnia różnicę między programowaniem strukturalnym i obiektowym, rozumie sytuacje, w których programowanie obiektowe jest koniecznością	– wyjaśnia bardziej zaawansowane pojęcia: <i>dziedziczenie, polimorfizm</i>	– potrafi stosować w praktyce paradygmat obiektowy (czyli samodzielnie pisze obiektowo programy)
Jak rozwiązywać zadania programistyczne, czyli podstawowe porady na proste kody	– rozumie treść zadań programistycznych i umie przełożyć proste zadania na implementację w wybranym języku programowania	– świadomie stosuje zasady czytelnego pisania kodu (wcięcia, nazwy zmiennych etc.)	- stosuje dobre praktyki programistyczne (unikanie kopiuj-wklej, wydzielenie do podprocedur) - umie przetestować swój program, w tym napisać dla niego proste dane testowe	– pisze programy w sposób czytelny i zrozumiały dla innych – potrafi swobodnie testować program, wyszukiwać i poprawiać błędy	– rozwiązuje zadania na poziomie olimpijskim np. na serwerach typu „online judge”
<b>II. Arkusz kalkulacyjny</b>					
Podstawowe typy danych, czyli co możemy wpisać w komórkę arkusza	– uzasadnia wybranie odpowiedniego typu formatowania do podanych danych – rozróżnia sposoby adresowania	– potrafi używać podstawowych funkcji arkusza i stosować w nich różne typy adresowania – potrafi dostosować typ formatowania do podanych danych	– potrafi samodzielnie używać różne typy adresowania	– samodzielnie potrafi dobrać rodzaje adresowania, tak aby formuła była jak najbardziej uniwersalna i używać zagnieżdżonych formuł	

Trójkąt Sierpińskiego, czyli do czego przydaje się adresowanie względne i adresowanie bezwzględne oraz wypełnienie serią danych	– wie, jak wykorzystać podane przez nauczyciela odwzorowania do wygenerowania trójkąta Sierpińskiego	– potrafi samodzielnie wypełnić komórki podobnymi wartościami (liczby, daty)	– potrafi samodzielnie otworzyć pokazane ćwiczenie, rozumie, jak użyć podanego układu iterowanych rozwiązań	– potrafi samodzielnie generować inne fraktale niż te pokazane na lekcji (po podaniu mu wzoru na iterowane przekształcenia)	– rozumie i potrafi wyjaśnić, na czym polega generowanie fraktali za pomocą układu iterowanych rozwiązań
Jak korzystać z gotowych funkcji, podstawowe funkcje tekstowe, czyli jak łatwo modyfikować dane tekstowe	– zna i potrafi użyć funkcji pokazanych na lekcji	– potrafi przeczytać opis nowej funkcji i użyć jej w zadaniu	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach zagnieżdżonych	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach zagnieżdżonych, tak aby te formuły były uniwersalne
Jak korzystać z gotowych funkcji, podstawowe funkcje dotyczące daty i czasu, czyli jak łatwo pracować z datą.	– zna i potrafi użyć funkcji pokazanych na lekcji	– potrafi przeczytać opis nowej funkcji i użyć jej w zadaniu	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach zagnieżdżonych	– potrafi samodzielnie używać podanych funkcji i używać ich w formułach zagnieżdżonych, tak aby te formuły były uniwersalne
Jak szukać w Excelu, czyli gdzie to jest	– potrafi podać, jakie funkcje w arkuszu pozwalają wyszukiwać, potrafi używać funkcji WYSZUKAJ na prostych przykładach	– odtwarza przykłady użycia funkcji WYSZUKAJ podane w podręczniku	– potrafi użyć funkcji WYSZUKAJ.PIONOW O dla przykładów podanych w podręczniku	– potrafi samodzielnie stosować funkcję WYSZUKAJ dla dowolnych przykładów podanych przez nauczyciela i potrafi sam wskazać takie przykłady	– potrafi samodzielnie stosować funkcję WYSZUKAJ.PIONOWO dla dowolnych przykładów i potrafi sam wskazać takie przykłady
Co ukrywa numer PESEL, czyli do czego przydaje się adres mieszany	– potrafi z podanego numeru PESEL odczytać datę urodzin i płeć	– potrafi samodzielnie napisać formułę, która policzy z podanego numeru PESEL datę urodzin	– samodzielnie potrafi napisać formułę, która wyznaczy płeć z numeru PESEL	– potrafi napisać formułę, która sprawdzi czy numer PESEL jest prawidłowy	– samodzielnie potrafi napisać uniwersalne formuły wyznaczające datę urodzenia, płeć i sprawdzające poprawność dla wielu danych
Tabela przestawna, czyli wygodne grupowanie danych	– wie, kiedy stosować tabele przestawne, potrafi odczytywać informacje z tabeli przestawnej	– potrafi odtworzyć przykłady użycia tabeli przestawnej pokazane na lekcji	– samodzielnie potrafi utworzyć tabelę przestawną i policzyć, sumę, średnią, minimum, maksimum dla wyznaczonych grup	– potrafi samodzielnie utworzyć tabelę przestawną, zastosować w niej filtr i posortować dane według podanego klucza	– potrafi samodzielnie utworzyć tabele przestawne i dla grup tworzyć podgrupy oraz użyć funkcji liczących dla grup i podgrup

Wykresy, czyli jak atrakcyjnie przedstawiać dane	- potrafi odczytywać dane z wykresu - wie, jakie są podstawowe typy wykresu - potrafi dobrać odpowiedni rodzaj wykresu do podanych danych	- potrafi odtworzyć przykłady użycia tabeli przestawnej pokazane na lekcji	- samodzielnie potrafi utworzyć wykres i sformatować go według podanych wytycznych	- potrafi samodzielnie dobrać wykres do podanych danych i sformatować go tak, aby był czytelny – potrafi przedstawić kilka serii danych na wykresie	- potrafi utworzyć wykres z tabeli przestawnej i sformatować go, aby dane były czytelne
Pobieranie danych z pliku, czyli z notatnika do Excela.	- potrafi pobrać dane z pliku tekstowego zawierające tekst i liczby całkowite	- potrafi pobrać dane z pliku zawierające długie liczby i numer pesel	- potrafi pobrać dane zawierające liczby rzeczywiste	- potrafi pobrać dane z pliku zawierające datę i czas	- potrafi pobrać dane z dowolnego pliku zawierającego tekst ze znakami polskimi

### Ocena celująca

Uczeń:

- zna wymagane pojęcia i terminologię komputerową;
- posiada wymaganą na tym etapie nauczania przedmiotu wiedzę teoretyczną;
- perfekcyjnie i z dużą swobodą posługuje się oprogramowaniem komputerowym, wykorzystując opcje o wysokim stopniu trudności;
- perfekcyjnie i z dużą swobodą posługuje się usługami internetowymi, w tym chmurą;
- samodzielnie rozwiązuje przedstawione na zajęciach problemy informatyczne;
- wykonuje ćwiczenia, prace i projekty z dużym stopniem samodzielności i własnej inwencji, złożonością oraz bogactwem użytych efektów i opcji, pomysłowością, oryginalnością, a także wysokimi walorami estetycznymi;
- do swoich prac pozyskuje materiał z bardzo różnych źródeł wiedzy;
- wyróżnia się starannością i solidnością podczas wykonywania powierzonych zadań oraz aktywnością na lekcjach;
- przestrzega norm obowiązujących w pracowni komputerowej, internetowej netykiety, a także zasad związanych z przestrzeganiem praw autorskich;
- wykazuje ponadprzeciętne zainteresowanie przedmiotem, objawiające się poszerzoną wiedzą i umiejętnościami zdobywanymi na kółku informatycznym i we własnym zakresie;
- uczestniczy z dobrym wynikiem w szkolnych i międzyszkolnych konkursach informatycznych, grafiki komputerowej, animacji itp.;
- układa algorytmy i programy komputerowe zawierające własne rozwiązania problemów programistycznych.

### Ocena bardzo dobra

Uczeń:

- zna i rozumie wszystkie wymagane pojęcia i terminologię komputerową;
- posiada wymaganą na tym etapie nauczania przedmiotu wiedzę teoretyczną;
- posługuje się oprogramowaniem komputerowym, również większością opcji o wysokim stopniu trudności;
- posługuje się usługami internetowymi;
- samodzielnie rozwiązuje prostsze problemy informatyczne, układając samodzielnie algorytmy i programując je w języku programowania, np. C++;
- wykonuje ćwiczenia, prace i projekty z dużą starannością i dokładnością w odtworzeniu zaprezentowanego przez nauczyciela wzoru czy przykładu.

### **Ocena dobra**

Uczeń:

- zna i rozumie w dużym zakresie wymagane pojęcia i terminologię komputerową;
- posiada niewielkie braki w wiedzy teoretycznej;
- z niewielkimi potknięciami posługuje się oprogramowaniem komputerowym, zna dużo opcji w nich zawartych, również częściowo tych o dużym stopniu trudności;
- sprawnie posługuje się usługami internetowymi;
- wykonuje ćwiczenia, prace i projekty z niewielkimi brakami w stosunku do przedstawionego przez nauczyciela wzoru lub przykładu;
- poprawnie używa IDE i układa samodzielnie proste programy;
- trudniejsze zadania rozwiązuje z niewielką pomocą nauczyciela lub tutoriala;
- aktywnie korzysta z podręcznika.

### **Ocena dostateczna**

Uczeń:

- nie wykazuje szczególnego zainteresowania przedmiotem, niemniej zadane ćwiczenia i prace stara się, mimo trudności, wykonać jak najlepiej i w określonym czasie;
- w posiadanej wiedzy teoretycznej prezentuje duże braki, niemniej najważniejsze elementy materiału ma opanowane;
- z niewielką pomocą nauczyciela posługuje się oprogramowaniem komputerowym oraz usługami internetowymi;
- ćwiczenia, prace i projekty wykonuje bez należytej dbałości, bez stosowania wielu opcji i efektów;
- używa IDE, często korzystając z opcji Pomoc;
- układa programy i algorytmy z dużą pomocą nauczyciela i podręcznika;
- trudniejsze zadania rozwiązuje, w dużym stopniu wzorując się na gotowych opracowaniach;
- rozumie zastosowania informatyki w różnych dziedzinach po ich omówieniu przez nauczyciela.

### **Ocena dopuszczająca**

Uczeń:

- nie wykazuje zainteresowania przedmiotem;
- posiada minimalny wymagany zasób wiedzy teoretycznej;
- z pomocą nauczyciela, często niezbyt chętnie, posługuje się oprogramowaniem komputerowym, wykorzystując tylko najbardziej podstawowe, wybrane opcje i efekty;
- z dużą pomocą nauczyciela posługuje się usługami internetowymi;
- ćwiczenia, prace i projekty wykonuje niestarannie, z dużymi brakami w stosunku do zaprezentowanych przez nauczyciela zasad lub przykładów, z wykorzystaniem najprostszych opcji i narzędzi;
- nie umie w pełni samodzielnie posługiwać się IDE;
- z pomocą nauczyciela układa bardzo proste programy komputerowe;
- nie umie wytłumaczyć działania gotowych algorytm