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

Uwaga!

W planie pominięto podstawowe umiejętności, które uczeń powinien już posiąść wcześniej, np. zachowywanie plików projektów, wczytywanie dokumentów do edycji i posługiwanie się systemem operacyjnym.

Kryteria danej oceny opracowano, zakładając, że zostały spełnione kryteria ocen niższych.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Temat** | **Ocena dopuszczająca.****Uczeń:** | **Ocena dostateczna****Uczeń:** | **Ocena dobra****Uczeń:** | **Ocena bardzo dobra****Uczeń:** | **Ocena celująca****Uczeń:** |
| **I. Algorytmika i programowanie** |
| Więcej o liczbach, czyli logarytm, sito Eratostenesa i schemat Hornera. | – zna definicję logarytmu– zna ogólną ideę algorytmu sita Eratostenesa (wykreślanie wielokrotności kolejnych liczb pierwszych)– potrafi zapisać naiwny („pierwiastkowy”) algorytm sprawdzania pierwszości | – zna przykłady sytuacji, w których używa się logarytmu– potrafi prześledzić działanie algorytmu sita Eratostenesa na konkretnych danych | – potrafi rozpoznać złożoność logarytmiczną w algorytmach– potrafi zapisać w pseudojęzyku schemat Hornera | – potrafi zaimplementować sito Eratostenesa i schemat Hornera w wybranym języku programowania– zna złożoność algorytmu sita Eratostenesa i porównuje ją z „pierwiastkowym” algorytmem | – |
| Rekursja raz jeszcze, czyli dziel i zwyciężaj. | – potrafi zasymulować (prześledzić działanie) prostej funkcji rekurencyjnej | – zna warunki działania rekursji, potrafi przekształcać proste algorytmy (z jednym wywołaniem rekurencyjnym) z iteracyjnych na rekurencyjne i odwrotnie | – prawidłowo analizuje algorytmy rekurencyjne typu „dziel i zwyciężaj”, zna i umie stosować warunki działania rekursji w projektowaniu algorytmów | – potrafi samodzielnie zaprojektować algorytm opierający się na rekursji z dwoma wywołaniami typu „dziel i zwyciężaj” | – potrafi zanalizować nietypowe wywołania rekurencyjne (np. na więcej niż dwóch podzadaniach albo na różniących się od siebie podzadaniach) |
| Sortowanie przez scalanie, czyli pierwszy efektywny algorytm sortowania. | – zna ogólną ideę algorytmu sortowania przez scalanie (podzielić na dwie połowy, rekursja, scalenie) | – zna ogólną ideę scalania, potrafi podać złożoność algorytmu (n log n) | – potrafi zapisać cały algorytm sortowania przez scalanie w pseudojęzyku | – potrafi zapisać algorytm sortowania przez scalanie w wybranym języku programowania bez błędów i pilnując przypadków brzegowych | – potrafi uargumentować złożoność algorytmu sortowania przez scalanie |
| Sortowanie szybkie, czyli jak się sortuje w praktyce. | – zna ogólną ideę algorytmu szybkiego sortowania (przestawić elementy, a potem wywołanie rekurencyjne) | – wie, czym jest klucz i zna ideę operacji przestawiania, zna złożoność algorytmu w średnim wypadku | – potrafi zapisać cały algorytm sortowania przez scalanie w pseudojęzyku | – potrafi zapisać algorytm sortowania przez scalanie w wybranym języku programowania, rozumie korzyść z losowania klucza | – potrafi przedyskutować i uargumentować złożoność algorytmu przy różnych wyborach klucza (najlepszym/najgorszym) |
| Specjalne elementy w tablicy, czyli największy, najmniejszy i lider. | – potrafi znaleźć oddzielnie element najmniejszy i największy– wie, czym są 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. std::vector, std::list, 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 | – 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 obiektowgo („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:~~ *~~dziedziczenie~~*~~,~~ *~~polimorfizm~~* | ~~– 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, wydzielanie 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” |
| **II. Arkusz kalkulacyjny** |
| 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 różnych typów 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 wyszukać, potrafi używać funkcji WYSZUKAJ na prostych przykładach | – odtwarza przykłady użycia funkcji WYSZUKAJ podane w podręczniku | – potrafi użyć funkcji WYSZUKAJ.PIONOWO 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 przestawną 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 |