

I. Postanowienia ogólne

Opracowanie na podstawie:

1. Rozporządzenia Ministra Edukacji Narodowej w sprawie warunków i sposobu oceniania, klasyfikowania i promowania uczniów w szkołach oraz przeprowadzania sprawdzianów i egzaminów w szkołach publicznych;
2. Programu nauczania informatyki,
3. Podstawy programowej kształcenia ogólnego z informatyki
4. Statutu szkoły.

II. Przedmiotem oceny są

- wiedza i umiejętności wynikające z podstawy programowej nauczania informatyki oraz wymagań programu nauczania;
- aktywność i systematyczność
- wysiłek wkładany przez ucznia

III. Ocenie podlegają

- Sprawdziany z zakresu materiału programowego obejmującego więcej niż trzy ostatnie lekcje, zapowiedziane z tygodniowym wyprzedzeniem. Uczeń ma prawo do poprawy sprawdzianu. Poprawę uczeń może napisać w przeciągu dwóch tygodni od daty otrzymania oceny.
- Kartkówki, które nie są zapowiadane i obejmują trzy ostatnie lekcje
- Prace domowe
- Odpowiedzi ustne
- Aktywność na lekcji
- Praca w grupie

IV. Kryteria i sposoby oceniania

1. W pisemnych formach sprawdzania wiadomości i umiejętności ustala się następujące kryteria ocen wyrażone w procentach:

Stopień celujący	96% i powyżej
Stopień bardzo dobry	Między 86-95%
Stopień dobry	Między 71-85%
Stopień dostateczny	Między 51-70%
Stopień dopuszczający	Między 40-50%
Stopień niedostateczny	Poniżej 40%

V. Wymagania dotyczące egzaminu podwyższającego ocenę

- Wymogiem do złożenia podania o egzamin podwyższający ocenę jest konieczność napisania przez ucznia wszystkich wymaganych sprawdzianów

VI. Szczegółowe zasady okresowego podsumowania osiągnięć edukacyjnych

- Na początku roku szkolnego uczniowie zostają poinformowani przez nauczyciela o zakresie wymagań z informatyki, obowiązującym w danym roku (zakres wiadomości i umiejętności, które należy mieć opanowane na koniec roku szkolnego) oraz o sposobie i zasadach oceniania z danego przedmiotu.
- Nieobecność ucznia na lekcji zobowiązuje go do uzupełniania materiału we własnym zakresie
- Na lekcji uczeń może być oceniony za pracę na lekcji: odpowiedź, aktywność, wykonywane ćwiczenia lub brak pracy
- Korzystanie przez ucznia w czasie sprawdzianów, kartkówek i innych form sprawdzania wiedzy z niedozwolonych przez nauczyciela pomocy stanowi podstawę do wystawienia oceny niedostatecznej
- Uczeń, który był nieobecny na sprawdzianie, zalicza go w terminie ustalonym z nauczycielem
- Kartkówki nie podlegają poprawie
- Ocenę semestralną i roczną nauczyciel wystawia na podstawie ocen cząstkowych uzyskanych przez ucznia
- Uczeń jest zobowiązany przygotować się do lekcji z 3 ostatnich tematów
- Uczeń może dwa razy na semestr skorzystać z „nieprzygotowania” i nie być brany pod uwagę przy wyborze do odpowiedzi ustnej.
- Uczeń może uzyskiwać za aktywność na lekcji „plusy”. Po zebraniu odpowiedniej liczby plusów są one kasowane a uczeń otrzymuje ocenę celującą z aktywności.

Wymagania edukacyjne niezbędne do otrzymania przez ucznia poszczególnych śródrocznych i rocznych ocen klasyfikacyjnych z in-formatyki w klasie drugiej i trzeciej

Wymagania na oceny śródroczne klasy drugiej (I półrocze) obejmują wymagania z działów od I do II włącznie, zaś na oceny roczne obejmują wszystkie wymagania z działów od I do III włącznie (cały rok szkolny). Tematy z działu III są też wymagane w klasie trzeciej.

W związku z uszczupleniem przez MEN podstawy programowej, w rozkładzie materiału zmniejszyła się liczba godzin na realizację obowiązkowych zagadnień. Uzyskane w ten sposób dodatkowe godziny pozostają do dyspozycji nauczyciela w trakcie roku szkolnego. Zgodnie z założeniami MEN: *Ograniczony zakres treści nauczania – wymagań szczegółowych – da nauczycielom i uczniom więcej czasu na spokojniejszą i bardziej dogłębną realizację programów nauczania.*

Lp.	Temat	Liczba godzin	Zapisy podstawy programowej
Rozdział 1. Algorytmy na liczbach całkowitych i tekstach			
1	Od problemu do programu	3	I.1, I.2, I.3, RI.3, II.1, RII.2, RIIII.1
2	Systemy liczbowe i reprezentacja danych w komputerze	3	I.1, I.3, RI.7, RI.8, II.1, RII.2
3	Algorytmy zamiany reprezentacji liczb między systemami liczbowymi	3	I.1, I.2, I.2a, I.3, RI.5, RI.6, II.1, RII.2, RIIII.1, RIIII.1i
4	Czy to jest palindrom?	2	I.1, I.2b, I.3, RI.1, II.1, RII.2
5	Czy ta liczba jest pierwsza?	3	I.1, I.2, I.2a, I.3, RI.2, RI.3, RI.5, II.1, RII.2, RIIII.1, RIIII.2a
6	Działania na liczbach w systemach innych niż dziesiętny	3	I.1, I.2, I.3, RI.3, RI.6, RI.8, II.1, RII.2, RIIII.1, RIIII.2b
7	Algorytm Euklidesa i działania na ułamkach	3	I.1, I.2, I.2a, I.3, RI.2, RI.3, RI.5, RI.6, RI.10, II.1, RII.2, RIIII.1, RIIII.1a
8	Szyfr Cezara i inne szyfry podstawieniowe	3	I.1, I.2b, I.3, RI.3, II.1, RII.2
W	Wiesz, umiesz, zdasz	4	I.1, I.2, I.2a, I.2b, I.3, RI.2, RI.3, RI.5, RI.7, II.1, RIIII.1
Rozdział 2. Rozwiązywanie problemów z wykorzystaniem struktur danych			
9	Łamiemy szyfr Cezara	3	I.1, I.2b, I.3, RI.3, RI.10, II.1, RII.2
10	Poszukujemy liczby	2	I.1, I.2, I.3, RI.1, RI.2, RI.3, RI.4, RI.5, RI.6, RI.10, II.1, RII.2, RIIII.1, RIIII.1b, RIIII.1d, RIIII.3a, RIIII.3c
11	Jak ocenić złożoność obliczeniową algorytmu?	2	RI.2, RI.5, RI.6, RI.10,

Lp.	Temat	Liczba godzin	Zapisy podstawy programowej
12	Metody sortowania prostego	3	I.1, I.2, I.2c, I.3, RI.2, RI.3, RI.5, RI.6, RI.10, II.1, RII.2, RIIII.1, RIIII.3a
13	Sito Eratostenesa	2	I.1, I.3, RI.3, RI.5, RI.6; RI.10, II.1, RII.2, RIIII.1c
14	Szukamy różnych podciągów	4	I.1, RI.2, I.3, RI.3, RI.5, RI.6, RI.10, II.1, RII.2, RIIII.2c
W	Wiesz, umiesz, zdasz	4	I.1, I.2, I.2a, I.2b, I.2c, I.3, RI.3, RI.4, II.1, RII.2, RIIII.2c, RIIII.3a
Rozdział 3. Metody algorytmiczne			
15	Iteracja a rekurencja	4	I.1, I.3, RI.1, RI.2, RI.3, RI.4, RI.5, RI.10, II.1, RII.2, RIIII.1a, RIIII.1i, RIIII.3b
16	Metoda zachłanna	4	I.1, I.3, RI.1, RI.2, RI.3, RI.4, RI.10, II.1, RII.2, RIIII.3d
17	Programowanie dynamiczne	5	I.1, I.3, RI.2, RI.3, RI.4, RI.5, RI.10, II.1, RII.2, RIIII.2c, RIIII.3e
18	Dziel i zwyciężaj, czyli sortujemy sprawniej	4	I.1, I.3, RI.1, RI.2, RI.3, RI.4, RI.5, RI.6, RI.10, II.1, RII.2, RIIII.1e, RIIII.3b, RIIII.3c
W	Wiesz, umiesz, zdasz	4	I.1, I.2b, I.3, RI.3, RI.4, II.1, RII.2
P	Programowanie zespołowe – projekt zespołowy	3	I.1, I.3, RI.3, RI.4, RI.5, RI.10, II.1, II.2, RII.1, RII.2, RII.3b, RIV.1
Suma godzin: 71			

Plan wynikowy (propozycja)

Lp.	Temat	Liczba godzin	Osiągnięcia uczniów	
			Wymagania podstawowe. Uczeń:	Wymagania ponadpodstawowe. Uczeń:
1	Od problemu do programu	3	definiuje pojęcie specyfikacja algorytmu, określa dane i wyniki planuje kolejne kroki rozwiązania problemu omawia różne sposoby przedstawiania algorytmów (opis słowny, lista kroków, pseudokod) programuje i testuje rozwiązanie problemu sprawdza działanie algorytmów dla różnych danych tworzy algorytmy działania na liczbach całkowitych stosuje w języku C++ podstawowe konstrukcje programistyczne (operacje wejścia i wyjścia, instrukcja warunkowa, operatory matematyczne i logiczne) tworzy w języku C++ programy wykonujące działania na liczbach całkowitych	dobiera struktury i typy danych do rodzaju problemu wyszukuje optymalne rozwiązania problemów ocenia efektywność algorytmu objaśnia dobrany do danego problemu algorytm, uzasadnia jego poprawność i wybór
2	Systemy liczbowe i reprezentacja danych w komputerze	3	definiuje pojęcie pozycyjnego systemu liczbowego wymienia systemy liczbowe stosowane w informatyce definiuje pojęcia bit i bajt dokonuje konwersji między pozycyjnymi systemami liczbowymi, wykorzystując przy tym zależności między systemami binarnym i ósemkowym oraz binarnym i heksadecymalnym omawia sposób reprezentowania liczb całkowitych w komputerze wymienia typy danych służące do zapisu liczb całkowitych (short int, int, long int, long long int, unsigned), stosuje je w pisanych programach opisuje, jak w komputerze reprezentowane są znaki i napisy (char, string), odwołuje się do znaku w napisie za pomocą indeksu wyjaśnia, czym jest tablica kodów ASCII omawia działanie operacji logicznych i reprezentację ich	wykonuje zadania o podwyższonym stopniu trudności: oznaczone trzema gwiazdkami w podręczniku, z arkuszy maturalnych z lat poprzednich lub konkursów i olimpiad informatycznych

			wyników w komputerze (wynik może przyjmować wartość prawda – 1 lub fałsz – 0, co zajmuje 1 bajt pamięci)	
3	Algorytmy zamiany reprezentacji liczb między systemami liczbowymi	3	tworzy programy do konwersji między liczbami w systemach binarnym i dziesiętnym pisze programy konwertujące liczbę dziesiętną na liczbę w podanym systemie pozycyjnym posługuje się środowiskiem programistycznym, strukturami danych oraz językiem programowania w stopniu umożliwiającym implementację omawianych algorytmów stosuje binarną reprezentację liczby w algorytmie szybkiego podnoszenia do potęgi	pisze programy zamieniające liczby z systemu dziesiętnego na system heksadecymalny pisze programy o podwyższonym stopniu trudności z wykorzystaniem algorytmów zamiany: z zadań oznaczonych trzema gwiazdkami w podręczniku, z arkuszy maturalnych, z konkursów i olimpiad informatycznych posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych uzasadnia poprawność zaproponowanego rozwiązania korzysta z dostępnych bibliotek w tworzonych przez siebie programach tworzy własne funkcje rozwiązujące problemy
4	Czy to jest palindrom?	2	definiuje pojęcie palindromu określa, czy dany napis lub liczba są palindromami wykonuje operacje na napisach (wczytywanie napisów ze spacjami, sprawdzanie długości napisu, zamiana liter dużych na małe i odwrotnie, porównywanie znaków, znajdowanie oraz usuwanie fragmentów napisów) definiuje własne funkcje w języku C++, wyjaśnia celowość ich stosowania, rozróżnia parametry formalne i aktualne realizuje w języku C++ algorytmy sprawdzające, czy dany napis jest palindromem, oraz wyszukujące palindromy w zdaniach opisuje popularne funkcje oraz metody stosowane dla zmiennych typu string (toupper, tolower, size, substr, erase)	pisze programy dotyczące palindromów o podwyższonym stopniu trudności: z zadań oznaczonych trzema gwiazdkami w podręczniku, z arkuszy maturalnych, z konkursów i olimpiad informatycznych posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym optymalizuje algorytmy i ocenia ich efektywność
5	Czy ta liczba jest pierwsza?	3	wymienia podstawowe własności liczb pierwszych sprawdza, czy dana liczba jest pierwsza, stosując algorytm naiwny rozkłada liczbę złożoną na czynniki pierwsze	implementuje algorytmy dotyczące liczb pierwszych o podwyższonym stopniu trudności: z zadań oznaczonych trzema gwiazdkami w podręczniku, z arkuszy maturalnych, z konkursów i olimpiad informatycznych

			wyznacza liczby bliźniacze	posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym stosuje optymalny algorytm sprawdzający, czy liczba jest pierwsza, wykorzystując funkcję logiczną; uzasadnia jego efektywność pisze program rozkładający liczbę złożoną na sumę dwóch liczb pierwszych (hipoteza Goldbacha)
6	Działania na liczbach w systemach innych niż dziesiętny	3	wykonuje działania arytmetyczne na liczbach w różnych systemach pozycyjnych wykonuje obliczenia na dowolnie dużych liczbach, wykorzystując napisy wyjaśnia różnicę między operacjami na liczbach o podstawie od 1 do 9 i większej od 10 stosuje odejmowanie w dzieleniu pisemnym liczb binarnych stosuje dodawanie liczby przeciwnej zapisanej w kodzie U2 przy odejmowaniu liczby binarnej	wykonuje działania o podwyższonym stopniu trudności pisze programy wykonujące operacje arytmetyczne na liczbach w różnych systemach pozycyjnych optymalizuje programy, szacuje ich efektywność
7	Algorytm Euklidesa i działania na ułamkach	3	opisuje geometryczną interpretację algorytmu Euklidesa pisze program realizujący algorytm Euklidesa w wersjach z dzieleniem i odejmowaniem, stosując funkcję typu void stosuje strukturę do reprezentacji liczb wymiernych wykorzystuje algorytm Euklidesa do działań na ułamkach stosuje zmienne lokalne i globalne, a także przekazywanie parametrów przez wartość	pisze programy o podwyższonym stopniu trudności prezentujące zastosowanie algorytmu Euklidesa posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym stosuje funkcje i dobiera sposób przekazywania parametrów, jednocześnie go uzasadniając
8	Szyfr Cezara i inne szyfry podstawieniowe	3	definiuje szyfry: podstawieniowy, monoalfabetyczny i permutacyjny, wymienia przykłady takich szyfrów pisze program szyfrujący informację szyfrem Cezara z wykorzystaniem liter z polskimi znakami diakrytycznymi omawia szyfr Vigenere'a stosuje w swoich programach operacje plikowe – wczytywanie danych z pliku dyskowego, zapis wyniku do pliku	pisze programy szyfrujące o podwyższonym stopniu trudności posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym wykorzystuje odpowiednio dobrane struktury danych korzysta z funkcji bibliotecznych tworzy własne funkcje, dobierając sposób przekazywania parametrów
9	Łamiemy szyfr	3	wyjaśnia, na czym polega łamanie szyfru (kryptoanaliza)	opisuje różne sposoby łamania szyfrów i implementuje je

	Cezara		łamie szyfr Cezara, stosując analizę częstości stosuje algorytmy zliczające liczbę wystąpień znaków w tekście z zastosowaniem strukturalnego typu danych – tablic pisze program znajdujący maksimum w tablicy i wypisu- jący jego pozycję (algorytm „dziel i zwyciężaj”)	w języku C++ pisze programy deszyfrujące o podwyższonym poziomie trud- ności
10	Poszukujemy liczby	2	znajduje wartość w zbiorach uporządkowanym i nieupo- rządkowanym, stosując odpowiednio algorytmy wyszu- kiwania liniowego, liniowego z wartownikiem i binarnego pisze programy wykorzystujące przekazywanie paramet- ru do funkcji przez wskaźnik i referencję stosuje algorytm „dziel i zwyciężaj” do jednoczesnego znajdowania maksimum i minimum w zbiorze	pisze programy o podwyższonym stopniu trudności szacuje złożoność czasową zastosowanych algorytmów wyszu- kiwania wyjaśnia na przykładach różnice między różnymi sposobami przekazywania parametrów do funkcji podaje wzór na liczbę wykonywanych operacji w algorytmie „dziel i zwyciężaj”
11	Jak ocenić złożoność obliczeniową algorytmu?	4	definiuje złożoność obliczeniową algorytmu szacuje złożoność czasową i pamięciową wyjaśnia, czym jest złożoność oczekiwana (średnia), op- tymistyczna i pesymistyczna	określa złożoność czasową i pamięciową algorytmów z zasto- sowaniem odpowiednich wzorów rozróżnia pojęcia algorytmu naiwnego i optymalnego ocenia efektywność algorytmów
12	Metody sortowania prostego	3	definiuje pojęcie sortowania, prawidłowo określając klucz i porządek sortowania definiuje pojęcia sortowania <i>in situ</i> i stabilnego stosuje metody sortowania prostego do sortowania liczb w zbiorze – bąbelkowe i przez wybieranie szacuje złożoność obliczeniową stosowanych algorytmów definiuje operacje kluczowe (dominujące) w algorytmach sortowania pisze programy realizujące poznane algorytmy sortowa- nia	pisze programy sortujące o podwyższonym stopniu trudności: sortowanie danych w plikach tekstowych, sortowanie struktur podaje przykłady sortowania prostego w życiu codziennym dobiera właściwe struktury danych definiuje własne funkcje do rozwiązywania problemów z wyko- rzystaniem algorytmów sortowania ocenia wpływ pierwotnego ułożenia danych w zbiorze na liczbę wykonywanych operacji
13	Sito Eratostenesa	2	opisuje algorytmy sprawdzające, czy liczba jest pierwsza omawia i stosuje algorytm sita Eratostenesa do wyszuki- wania liczb pierwszych w określonym przedziale liczo- wym określa złożoność obliczeniową algorytmu	pisze programy o podwyższonym stopniu trudności wykorzy- stujące sito Eratostenesa posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym optymalizuje algorytm, dążąc do minimalnej złożoności obli-

				czeniowej
14	Szukamy różnych podciągów	4	definiuje pojęcia podciągu oraz podciągu spójnego znajduje w zbiorze podciągi o różnych własnościach oblicza długość najdłuższego niemalejącego spójnego podciągu oraz liczbę jego elementów wymienia i stosuje różne algorytmy znajdowania maksymalnej sumy elementów spójnych podciągów, oceniając ich złożoność obliczeniową znajduje w zbiorze spójny podciąg o maksymalnej sumie i wypisuje jego elementy	pisze programy o podwyższonym stopniu trudności wyszukiujące spójne podciągi posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych
15	Iteracja a rekurencja	4	opisuje zasadę działania rekurencji implementuje w języku C++ algorytmy rekurencyjne, określa warunki brzegowe porównuje iteracyjne i rekurencyjne wersje algorytmów opisuje zasadę złotego podziału oblicza n -ty wyraz ciągu Fibonacciego metodami iteracyjną i rekurencyjną wyjaśnia, na czym polega rozszerzony algorytm Euklidesa, oraz implementuje go w języku C++	pisze programy o podwyższonym stopniu trudności, np. sprawdzanie hipotezy Collatza posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych uzasadnia wybór iteracji lub rekurencji do rozwiązania problemu szacuje złożoność czasową stosowanych algorytmów oblicza liczbę wykonywanych operacji w algorytmach rekurencyjnych
16	Metoda zachłanna	3	wyjaśnia, na czym polega metoda zachłanna, i wymienia przykłady jej stosowania implementuje następujące algorytmy zachłanne: problem kasjera (wydawania reszty minimalną liczbą nominałów), problem telewizyja/kinomana (optymalny harmonogram wykorzystania sali), wyszukiwanie optymalnej drogi ocenia przydatność zastosowanych algorytmów stosuje własne kryterium porównania w funkcji sort z biblioteki STL	pisze programy o podwyższonym stopniu trudności z wykorzystaniem algorytmów zachłannych, stosując rekurencję i algorytmy z nawrotami posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych objaśnia algorytm wybrany do rozwiązania problemu oraz ocenia jego efektywność i niedoskonałość
17	Programowanie	5	wyjaśnia, na czym polega metoda dynamiczna	pisze programy o podwyższonym stopniu trudności wykorzy-

	dynamiczne		implementuje optymalne algorytmy dotyczące problemu kasjera, telewidza, znajdowania drogi stosuje metodę dynamiczną do znajdowania najdłuższego wspólnego podciągu porównuje metody zachłanną i dynamiczną	stujące algorytmy dynamiczne posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych
18	Dziel i zwyciężaj, czyli sortujemy sprawniej	4	omawia metodę „dziel i zwyciężaj” oraz rekurencję wyjaśnia, na czym polega algorytm sortowania szybkiego oraz przez scalanie i implementuje je ocenia i porównuje złożoność czasową i obliczeniową algorytmów	pisze programy o podwyższonym stopniu trudności wykorzystujące metodę „dziel i zwyciężaj” oraz algorytmy sortowania posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym do rozwiązania problemu dobiera optymalny algorytm i struktury danych, a także korzysta z funkcji bibliotecznych
P	Programowanie zespołowe	3	wyjaśnia, czym jest dokumentacja projektu (projektowa, użytkownika, techniczna), bierze czynny udział w jej tworzeniu aktywnie uczestniczy w realizacji projektu przyjmuje różne role w zespole realizującym projekt prezentuje efekty wspólnej pracy	przyjmuje rolę lidera odpowiedzialnego za zespół i projekt przydziela zadania, nadzoruje pracę innych opracowując złożone problemy, posługuje się aplikacjami w stopniu zaawansowanym