Wymagania edukacyjne na poszczególne oceny szkolne z przedmiotu informatyka poziom rozszerzony klasa 3

Dla klas I Liceum Ogólnokształcącego im. Generała Józefa Bema w Ostrołęce

Klasa 3

Ocena śródroczna

Wymagania na poszczególne oceny				
Konieczne (ocena dopuszczająca)	**Podstawowe** (ocena dostateczna)	**Rozszerzajace** (ocena dobra)	**Dopelniające** (ocena bardzo dobra)	**Wykraczjące** (ocena celujaca)
2	**3**	**4**	**5**	**6**
1. Metody algorytmiczne				
Uczeń: • definiuje podstawowe pojęcia z algorytmiki i programowania: algorytm, program, warunek, iteracja, rekurencja, • wymienia sposoby reprezentacji algorytmów, • korzysta ze środowiska programistycznego: pisze w nim kod, kompiluje i uruchamia program, odczytuje i zapisuje pliki, • pisze programy o niewielkim stopniu trudności,	Uczeń: • przedstawia krótkie algorytmy w postaci listy kroków, opisu słownego, pseudokodu, schematu blokowego, • implementuje w języku C++ algorytmy rekurencyjne: obliczanie elementów ciągu Fibonacciego, wartości silni i potęgi, • omawia rozszerzony algorytm Euklidesa, • formułuje algorytm wydawania reszty minimalną liczbą monet, harmonogramu wykorzystania	Uczeń: • określa specyfikację algorytmu (dane, wynik), • pisze programy o różnym stopniu trudności, szacuje ich efektywność, • przedstawia omawiane algorytmy w postaci opisu słownego, listy kroków, schematu blokowego, pseudokodu, • dobiera typy danych do realizacji problemu, • stosuje zmienne typu unsigned w tworzonych programach,	Uczeń: • charakteryzuje sytuacje algorytmiczne, proponuje sposoby ich rozwiązania, • pisze programy o podwyższonym stopniu trudności: oznaczone trzema gwiazdkami w podręczniku, • optymalizuje rozwiązania, • stosuje zaawansowane funkcje środowiska i języka programowania (np. z biblioteki STL), • dobiera struktury danych i metody do rodzaju problemu,	Uczeń: • charakteryzuje skomplikowane sytuacje algorytmiczne, proponuje optymalne rozwiązanie sytuacji problemowej z zastosowaniem złożonych struktur danych i biblioteki STL języka C++, • pisze programy o wysokim stopniu trudności: rozwiązując zadania z olimpiad przedmiotowych, konkursów informatycznych lub oznaczone trzema gwiazdkami w podręczniku,

• korzysta z podstawowych funkcji języka: operacji wejścia i wyjścia, instrukcji warunkowych i iteracyjnych, gotowych funkcji bibliotecznych, • wczytuje dane z pliku tekstowego, zapisuje wyniki w pliku, • definiuje pojęcia iteracji i rekurencji, • omawia zasadę złotego podziału, • opisuje rozszerzony algorytm Euklidesa, • omawia metody zachłanne na przykładzie problemu kasjera, harmonogramu sali, pakowania plecaka i wyszukiwania drogi, • porównuje metody zachłanną i dynamiczną	sali, pakowania plecaka, znajdowania drogi metodami zachłanną i dynamiczną,	• porównuje algorytmy iteracyjne i rekurencyjne (liczbę wykonywanych operacji), szacuje ich złożoność czasową, • zapisuje w postaci programu rozszerzony algorytm Euklidesa, wyjaśnia jego działanie i zastosowanie, • stosuje metodę zachłanną w programach – problem kasjera, harmonogram wykorzystania sali, wyszukiwanie drogi, pakowanie plecaka,	• stosuje różne sposoby przekazywania parametrów do funkcji, uzasadnia ich użycie, • pisze funkcje typu logicznego, • szacuje złożoność obliczeniową programów, • wykorzystuje poznane algorytmy do rozwiązywania problemów nieomawianych na lekcjach, • pisze programy obliczające liczbę operacji przenoszenia krążków w problemie wież Hanoi, stosując iterację i rekurencję, • do implementacji rozszerzonego algorytmu Euklidesa stosuje zarówno iterację, jak i rekurencję, • stosuje metody zachłanną i dynamiczną w problemach kasjera, harmonogramu wykorzystania sali, pakowania plecaka i wyszukiwania drogi, wskazuje wady i zalety obu metod, szacuje złożoność czasową,	• implementuje w języku C++ algorytm Euklidesa, stosując iterację i rekurencję, • implementuje w języku C++ algorytm wyszukiwania binarnego w wersji rekurencyjnej, • pisze programy sortujące dane różnego typu w plikach tekstowych (liczby, napisy, pary), • stosuje zaawansowane algorytmy wyszukiwania, np. najlepszego wyboru (trwałych par), stosując rekurencję, • pisze programy obliczające liczbę operacji przenoszenia krążków w problemie wież Hanoi, stosując iterację i rekurencję, • stosuje w programach algorytmy sortowania inne niż omawiane na lekcjach (np. heapsort),

Wymagania na poszczególne oceny				
Konieczne (ocena dopuszczająca)	**Podstawowe** (ocena dostateczna)	**Rozszerzajace** (ocena dobra)	**Dopelniające** (ocena bardzo dobra)	**Wykraczjące** (ocena celujaca)
2	**3**	**4**	**5**	**6**
2. Rozwiązywanie problemów z wykorzystaniem dynamicznych struktur danych				
Uczeń: • pisze programy o niewielkim stopniu trudności, • wyjaśnia, co to jest notacja infiksowa, notacja prefiksowa, odwrotna notacja polska, drzewo wyrażenia algebraicznego, • definiuje pojęcie dynamicznej struktury danych, • definiuje dynamiczne struktury danych takie jak: stos, kolejka, lista, `vector`, • wymienia rodzaje list, • wyjaśnia, na czym polega sortowanie leksykograficzne, • definiuje graf, wymienia elementy i rodzaje grafów, wymienia sposoby reprezentacji grafu (macierz sąsiedztwa, lista sąsiedztwa),	Uczeń: • wyróżnia operacje, które można wykonywać na dynamicznych strukturach danych (stosie, kolejce, liście, typie `vector`), • omawia zastosowanie dynamicznych struktur danych na różnych przykładach, • zapisuje wyrażenia algebraiczne bez użycia nawiasów, w tym w postaci odwrotnej notacji polskiej, • oblicza wartość wyrażenia arytmetycznego zapisanego w odwrotnej notacji polskiej, • omawia algorytmy znajdowania wyjścia z labiryntu z wykorzystaniem iteracji i rekurencji, • symuluje problem Flawiusza, • sortuje dane leksykograficznie, • stosuje typ `vector` do reprezentacji grafu w postaci list sąsiedztwa,	Uczeń: • dobiera typy danych do rozwiązania problemu, • do przeglądania grafu stosuje algorytm przeszukiwania w głąb (DFS) oraz algorytm przeszukiwania grafu wszerz (BFS), • omawia algorytm Dijkstry,	Uczeń: • pisze programy o podwyższonym stopniu trudności: rozwiązuje zadania oznaczone trzema gwiazdkami w podręczniku, • optymalizuje rozwiązania, • stosuje zaawansowane funkcje środowiska i języka programowania, • dobiera struktury danych i metody do rodzaju problemu, • szacuje złożoność algorytmów, • implementuje algorytmy grafowe – BFS, DFS, algorytm Dijkstry,	Uczeń: • optymalizuje programy, szacuje ich efektywność, • wykorzystuje poznane algorytmy do rozwiązywania problemów nieomawianych na lekcjach, np. sprawdzanie spójności grafu

Ocenę niedostateczną otrzymuje uczeń, który nie opanował 80% wymagań na ocenę dopuszczającą.

Ocena roczna

Wymagania na poszczególne oceny				
Konieczne (ocena dopuszczająca)	**Podstawowe** (ocena dostateczna)	**Rozszerzajace** (ocena dobra)	**Dopelniające** (ocena bardzo dobra)	**Wykraczjące** (ocena celujaca)
2	**3**	**4**	**5**	**6**
3. Algorytmy numeryczne				
Uczeń: • omawia różnice między stałoprzecinkową a zmiennoprzecinkową reprezentacją liczb rzeczywistych w komputerze, • wymienia rodzaje błędów w obliczeniach komputerowych, rozróżnia błąd względny i bezwzględny, • znajduje wartość wielomianu algorytmem naiwnym, • wie, na czym polegają podstawowe metody obliczeń przybliżonych, • zna proste algorytmy badające własności geometryczne (np. położenie punktu względem prostej), • wyjaśnia, co to jest fraktal, wskazuje przykłady struktur fraktalnych występujących w przyrodzie	Uczeń: • wyjaśnia różnicę między przekazywaniem parametrów do funkcji przez wartość i przez referencję, • wykorzystuje pliki tekstowe do wczytywania danych i zapisywania wyników, • omawia algorytm znajdujący rozwinięcie binarne nieskracalnego ułamka właściwego, • zapisuje liczby w postaci znormalizowanej, • definiuje liczby pojedynczej precyzji i liczby podwójnej precyzji, • wykonuje działania na liczbach zmiennoprzecinkowych, • wskazuje różnice między algorytmem stabilnym a algorytmem niestabilnym,	Uczeń: • znajduje reprezentację liczby zapisanej w systemie dziesiętnym jako liczby pojedynczej i liczby podwójnej precyzji, • świadomie używa typów `float` i `double` w zadaniach, • stosuje schemat Hornera do zamiany liczby w systemie pozycyjnym o wybranej podstawie na liczbę dziesiętną, • stosuje metodę Monte Carlo w obliczeniach przybliżonych, • w algorytmach badających własności geometryczne wykorzystuje macierz oraz regułę Sarrusa do obliczania wyznacznika macierzy	Uczeń: • w reprezentacji liczb rzeczywistych w komputerze stosuje reprezentację stało- lub zmiennoprzecinkową zgodnie ze specyfikacją algorytmu, minimalizując błędy w obliczeniach, • stosuje schemat Hornera do szybkiego podnoszenia do potęgi, • implementuje algorytmy numeryczne: znajdowania miejsc zerowych funkcji oraz obliczania pierwiastka kwadratowego metodą bisekcji, obliczania pierwiastka kwadratowego metodą Newtona–Raphsona, obliczania pola obszaru zamkniętego metodą prostokątów i metodą trapezów, znajdowania przybliżenia liczby pi oraz symulacja ruchów Browna metodą Monte Carlo,	Uczeń: • protrafi napisac program implemetujacy metode Monte Carlo do obliczania pól figur w układzie 5spółrzędnych • potrafi napisac program sprawdzający przynależność punktu P do wielokąta wklęsłego, wykorzytuje do tego operacje na plikach • implementuje algorytm rysujący drzewo Pitagorasa stopnia n • implementuje algorytm rysujący kostke Mengera stopnia n

	• znajduje pierwiastki równania kwadratowego algorytmem stabilnym i algorytmem niestabilnym, • implementuje algorytm obliczający wartość wielomianu z zastosowaniem schematu Hornera, • stosuje w algorytmach numerycznych metody: bisekcji, Newtona–Raphsona, trapezów, prostokątów, • omawia algorytmy badające własności geometryczne – położenie punktu względem prostej, przecinania się odcinków, przynależności punktu do figury, • podaje przykłady fraktali (zbiór Cantora, drzewo binarne, dywan Sierpińskiego, płatek Kocha), wyjaśnia sposób tworzenia tych fraktali		• implementuje algorytmy badające własności geometryczne, • implementuje algorytmy generujące fraktale danego stopnia, • stosuje metodę IFS do tworzenia fraktali w arkuszu kalkulacyjnym	

Wymagania na poszczególne oceny				
Konieczne (ocena dopuszczająca)	**Podstawowe** (ocena dostateczna)	**Rozszerzajace** (ocena dobra)	**Dopelniające** (ocena bardzo dobra)	**Wykraczające** (ocena celujaca)
2	**3**	**4**	**5**	**6**
4. Zaawansowane algorytmy i techniki programistyczne				
• Uczeń: • wyszukuje wzorzec w tekście algorytmem naiwnym, • rozumie działanie funkcji haszującej, • wskazuje różnice między kryptografią symetryczną i kryptografią asymetryczną, definiuje pojęcia klucz publiczny i klucz prywatny, • wyjaśnia, do czego służy algorytm RSA, i wyróżnia główne etapy tego algorytmu (generowanie kluczy, szyfrowanie z kluczem publicznym oraz deszyfrowanie z kluczem prywatnym), • definiuje programowanie strukturalne,	• Uczeń: • implementuje algorytm naiwny wyszukiwania wzorca w tekście, • wyjaśnia metodę haszowania, • wyjaśnia, jak generuje się klucze publiczny i prywatny oraz szyfruje i deszyfruje informacje w algorytmie RSA, • wyjaśnia, na czym polegają metoda zstępująca i metoda wstępująca, • w programowaniu obiektowym definiuje własne klasy, korzystając ze specyfikatorów dostępu	• Uczeń: • omawia algorytm Karpa–Rabina do wyszukiwania wzorca w tekście z zastosowaniem funkcji haszującej, • pisze program generujący klucz prywatny i klucz publiczny w algorytmie RSA, • w programowaniu obiektowym stosuje hierarchię klas, wyjaśnia, na czym polega hermetyzacja danych i jakie jest zastosowanie operatora zasięgu	• Uczeń: • stosuje funkcję haszującą oraz algorytm Karpa–Rabina w programach wyszukujących wzorzec w tekście, • pisze programy szyfrujące i deszyfrujące informacje w algorytmie RSA, • stosuje programowanie obiektowe, definiując własne klasy, obiekty, atrybuty i metody, deklaruje konstruktory w klasach, wyjaśnia, na czym polega polimorfizm i czym są metody wirtualne,	Uczeń: • tworzy program implementujący algorytm Boyera – Moore'a • tworzy program implementujący algorytm Morrisa-Pratta do wyszukiwania wzorca w tekście • pisze program szyfrujący plik tekstowy algorytmem RSA stosując zadany klucz publiczny i wyznaczona liczbę do szyfrowania • pisze program deszyfrujący algorymem RSA plik tekstowy za pomocą zadanego klucza prywatnego • tworzy metody wirtualne dla klas • wykorzystuje metody wirualne klas w programach

Ocenę niedostateczną otrzymuje uczeń, który nie opanował 80% wymagań na ocenę dopuszczającą.

Legenda:

Ocena dopuszczająca wymagania [2]

Ocena dostateczna wymagania [2+3]

Ocena dobra wymagania [2+3+4]

Ocena bardzo dobra wymagania [2+3+4+5]

Ocena celująca wymagania [2+3+4+5+6]