Wymagania edukacyjne z informatyki

klasa 2

poziom rozszerzony

Przyjmujemy, że uczeń spełnia wymagania na ocenę wyższą, jeśli spełnia

jednocześnie wymagania na ocenę niższą oraz dodatkowe wymagania.

Proponujemy:

→Wymagania na ocenę dopuszczającą.

→Wymagania na ocenę dostateczną zawierają wymagania na ocenę dopuszczającą.

→Wymagania na ocenę dobrą zawierają wymagania na ocenę dostateczną i dopuszczającą.

→Wymagania na ocenę bardzo dobrą zawierają wymagania na ocenę dobrą, dostateczną i dopuszczającą.

→Wymagania na ocenę celującą zawierają wymagania na oceną bardzo dobrą, dobrą, dostateczną i dopuszczającą.

Rozdział 1. Społeczeństwo w internecie

Ocenę celującą otrzymuje uczeń, który:

- opisuje metody stosowane w kryptografii symetrycznej i asymetrycznej,
- wyjaśnia zasadę stosowania podpisu elektronicznego,

Ocenę bardzo dobrą otrzymuje uczeń, który:

- wyjaśnia, jak zwiększyć swoje bezpieczeństwo w sieci poprzez stosowanie różnych technik,
- Ocenę dobrą otrzymuje uczeń, który:
- wymienia zasady ochrony danych osobowych,
- opisuje zastosowania technologii komputerowej w różnych dziedzinach życia,
- opisuje rodzaje ataków sieciowych,
- Ocenę dostateczną otrzymuje uczeń, który:
- wymienia zasady komunikacji w sieci internet (netykieta),
- wymienia zagrożenia wynikające ze złej komunikacji w sieci,
- opisuje wpływ rozwoju technologii na zmiany w społeczeństwie,
- wymienia i opisuje rodzaje szkodliwego oprogramowania,

- Ocenę dopuszczającą otrzymuje uczeń, który:
- wyjaśnia pojęcia: wykluczenie i włączenie cyfrowe,
- podaje przykłady negatywnych zachowań w sieci internet,

Rozdział 2. Strony WWW i grafika komputerowa

Ocenę celującą otrzymuje uczeń, który:

- tworzy style opisujące wygląd strony WWW,
- dodaje do strony elementy odpowiedzialne za jej responsywność,
- buduje stronę z wykorzystaniem systemu CMS i publikuje ją w internecie,
- tworzy złożone modele 3D.

Ocenę bardzo dobrą otrzymuje uczeń, który:

- korzysta ze ścieżek względnych i bezwzględnych w kodzie HTML,
- poprawnie tworzy tabele o dowolnej strukturze,
- dołącza style kaskadowe do dokumentu HTML,
- tworzy ciekawą stronę WWW i publikuje ją w internecie,
- poprawnie używa narzędzia do rysowania krzywych Béziera,
- wycina dowolne elementy z obrazu rastrowego,
- tworzy w programach do grafiki wektorowej infografiki według wzoru,
- tworzy bryły obrotowe 3D na podstawie ich przekroju.

Ocenę dobrą otrzymuje uczeń, który:

- umieszcza zdjęcia na stronie WWW,
- tworzy linki do zasobów zewnętrznych oraz miejsc w obrębie jednej strony,
- poprawnie i na różne sposoby korzysta z opisu kolorów w języku HTML,
- wymienia podstawowe narzędzia programu GIMP,
- korzysta z warstw podczas pracy z programem GIMP,
- pracuje na warstwach w programie do grafiki wektorowej.

Ocenę dostateczną otrzymuje uczeń, który:

- opisuje podstawową strukturę strony w języku HTML,
- tworzy nagłówki w języku HTML,
- wstawia komentarze w kodzie HTML,
- tworzy listy uporządkowane i nieuporządkowane,
- rozumie cel pozycjonowania stron WWW,
- skaluje i kadruje obraz, dostosowując go do zadanego rozmiaru,
- wymienia podstawowe narzędzia programu Inkscape.

Ocenę dopuszczającą otrzymuje uczeń, który:

- zapisuje plik, nadając mu rozszerzenie .html,

- rozróżnia sekcje HEAD i BODY oraz opisuje różnicę między tymi częściami kodu,
- wymienia podstawowe znaczniki formatowania tekstu w języku HTML,
- opisuje budowę znacznika HTML,
- wyjaśnia pojęcie responsywności strony WWW,
- uruchamia stronę WWW na smartfonie,
- określa różnicę pomiędzy grafiką rastrową a wektorową,
- zapisuje wynik swojej pracy w różnych formatach graficznych,
- wyjaśnia, jak uruchomić środowisko do grafiki 3D online.

Rozdział 3. Rozwiązywanie problemów z wykorzystaniem struktur danych

Ocenę celującą otrzymuje uczeń, który:

- 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: z olimpiad przedmiotowych, konkursów informatycznych wyszukuje palindromy lub anagramy w plikach tekstowych,
- pisze program rozkładający liczbę złożoną na dwie liczby pierwsze (hipoteza Goldbacha),
- 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 i struktury danych do wyszukiwania spójnych podciągów,
- stosuje zaawansowane algorytmy wyszukiwania, np. najlepszego wyboru (trwałych par), stosując rekurencję,
- stosuje w programach algorytmy sortowania inne niż omawiane na lekcjach (np. heapsort),
- bierze udział w olimpiadach i konkursach, zajmując punktowane miejsca,
- w projektach zespołowych przyjmuje rolę lidera.

Ocenę bardzo dobrą otrzymuje uczeń, który:

- pisze program realizujący algorytm sita Eratostenesa,
- implementuje w języku C++ algorytmy wyszukujące spójne podciągi o różnych cechach,
- stosuje w programach algorytmy wyszukiwania lidera i idola w zbiorze,
- porównuje algorytmy iteracyjne i rekurencyjne (liczbę wykonywanych operacji), szacuje ich złożoność czasową,
- zapisuje w postaci programu rozszerzony algorytm Euklidesa, wyjaśnia jego działanie i zastosowanie, przy testowaniu liczby na pierwszość stosuje funkcję typu logicznego,
- wyszukuje liczby bliźniacze,
- wykorzystuje algorytm Euklidesa do działań na ułamkach, stosując struktury lub pary (typ pair),

- pisze program wyszukujący jednocześnie minimum i maksimum w zbiorze z wykorzystaniem metody „dziel i zwyciężaj” oraz podaje wzór na liczbę wykonywanych operacji,
- szacuje złożoność obliczeniową programów sortujących, modyfikuje funkcje sortujące, zmieniając porządek sortowania,
- wykorzystuje poznane algorytmy do rozwiązywania problemów nieomawianych na lekcjach,
- optymalizuje program realizujący algorytm sita Eratostenesa i szacuje jego złożoność czasową,
- pisze programy wyszukujące lidera i idola w zbiorze, optymalizuje je, szacuje złożoność czasową,

Ocenę dobrą otrzymuje uczeń, który:

- implementuje w języku C++ i optymalizuje algorytm sprawdzający, czy liczba jest pierwsza,
- pisze program rozkładający liczby na czynniki pierwsze,
- stosuje w programach algorytm Euklidesa do obliczenia NWD i NWW,

Ocenę dostateczną otrzymuje uczeń, który:

- przedstawia krótkie algorytmy w postaci listy kroków, opisu słownego, pseudokodu, schematu blokowego,
- implementuje w języku C++ algorytmy wyszukiwania liniowego i liniowego z wartownikiem, porównuje ich efektywność,
- przedstawia w postaci listy kroków lub schematu blokowego algorytmy sortowania prostego (bąbelkowe, przez wybieranie) oraz szybkiego i przez scalanie, określa operacje dominujące,
- omawia algorytm sita Eratostenesa,
- przedstawia algorytmy znajdowania spójnych podciągów, wyznaczania najdłuższego z nich oraz podciągu o największej sumie elementów,
- omawia algorytm znajdowania idola i lidera w zbiorze,
- implementuje w języku C++ algorytmy rekurencyjne: obliczanie elementów ciągu Fibonacciego, wartości silni i potęgi,
- omawia rozszerzony algorytm Euklidesa,
- omawia algorytm zliczania znaków w tekście oraz wyszukujący maksimum z wykorzystaniem tablic,

Ocenę dopuszczającą otrzymuje uczeń, który:

- omawia pojęcia: złożoność obliczeniowa algorytmu, algorytm naiwny, algorytm optymalny, złożoność pesymistyczna, złożoność oczekiwana (średnia),
- korzysta z podstawowych funkcji języka: operacji wejścia i wyjścia, instrukcji warunkowych i iteracyjnych, gotowych funkcji bibliotecznych,

- wymienia podstawowe typy danych, operacje arytmetyczne i logiczne,
- w pisanych programach korzysta ze strukturalnych typów danych: napisów, struktur, tablic,
- definiuje pojęcie systemów liczbowych,
- omawia metody sortowania prostego (bąbelkowe, przez wybieranie) oraz szybkiego i przez scalanie na przykładowych danych,
- wypisuje liczby pierwsze z zadanego przedziału, stosując metodę sita Eratostenesa,
- wyszukuje w ciągu liczb spójne podciągi (nierosnący, niemalejący, stały), wskazuje najdłuższe, oblicza ich sumę,
- wskazuje idola i lidera w zbiorze danych,
- definiuje pojęcia iteracji i rekurencji,
- omawia zasadę złotego podziału,
- opisuje rozszerzony algorytm Euklidesa,
- omawia algorytmy wyszukiwania liczby w zbiorach uporządkowanym i nieuporządkowanym,

Rozdział 4. Algorytmy na tekstach

Ocenę celującą otrzymuje uczeń, który:

- 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: z olimpiad przedmiotowych, konkursów informatycznych lub oznaczone trzema gwiazdkami w podręczniku,
- wyszukuje palindromy lub anagramy w plikach tekstowych,
- tworzy palindromy z napisów, dopisując minimalną liczbę znaków,
- implementuje w języku C++ algorytm wyszukiwania binarnego w wersji rekurencyjnej, pisze programy sortujące dane różnego typu w plikach tekstowych (liczby, napisy, pary),
- stosuje w programach algorytmy sortowania inne niż omawiane na lekcjach (np. heapsort),
- bierze udział w olimpiadach i konkursach, zajmując punktowane miejsca,
- w projektach zespołowych przyjmuje rolę lidera.

Ocenę bardzo dobrą otrzymuje uczeń, który:

- stosuje różne sposoby przekazywania parametrów do funkcji, uzasadnia ich użycie,
- pisze funkcje typu logicznego, np. sprawdzającą, czy napis jest palindromem,
- sprawdza, czy napisy są anagramami, stosując sortowanie lub zliczanie znaków,
- szyfruje dane wczytane z pliku z uwzględnieniem polskich znaków diakrytycznych,
- pisze program odczytujący informację ukrytą za pomocą szyfru Cezara z wykorzystaniem analizy częstości znaków w tekście,
- stosuje funkcję haszującą oraz algorytm Karpa–Rabina w programach wyszukujących wzorzec w tekście,

Ocenę dobrą otrzymuje uczeń, który:

- implementuje w języku C++ algorytmy sprawdzające, czy napis jest palindromem,
- pisze programy sprawdzające, czy dwa napisy są anagramami, wykorzystując funkcję sort z biblioteki STL,
- szyfruje dane wczytane z pliku tekstowego,
- implementuje w języku C++ algorytm zliczania znaków w tekście oraz wyszukujący maksimum z wykorzystaniem tablic,
- omawia algorytm Karpa–Rabina do wyszukiwania wzorca w tekście z zastosowaniem funkcji haszującej,

Ocenę dostateczną otrzymuje uczeń, który:

- korzysta z biblioteki string do operacji na łańcuchach znaków,
- wykonuje operacje na napisach, wykorzystując słowa kluczowe: size, find, substr, erase, toupper, tolower,
- wczytuje napisy ze spacjami, wykorzystując słowo kluczowe getline,
- tworzy algorytmy sprawdzające, czy napis jest palindromem,
- przedstawia w postaci algorytmu problem wyszukiwania anagramów,
- pisze program szyfrujący napis szyfrem Cezara,
- implementuje algorytm naiwny wyszukiwania wzorca w tekście,
- wyjaśnia metodę haszowania,

Ocenę dopuszczającą otrzymuje uczeń, który:

- wyjaśnia, czym jest tablica kodów ASCII,
- wyjaśnia, czym są palindrom i anagram, podaje przykłady,
- omawia szyfr Cezara jako przykład szyfru podstawieniowego i szyfr kolumnowy jako przykład szyfru przestawieniowego,
- omawia algorytm zliczania znaków w tekście,
- wczytuje dane z pliku tekstowego, zapisuje wyniki w pliku,
- wyszukuje wzorzec w tekście algorytmem naiwnym,
- rozumie działanie funkcji haszującej,

Rozdział 5. Algorytmy na liczbach całkowitych i rzeczywistych (reprezentacja liczb w komputerze)

Ocenę celującą otrzymuje uczeń, który:

- 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: z olimpiad przedmiotowych, konkursów informatycznych wyszukuje palindromy lub anagramy w plikach tekstowych,
- w projektach zespołowych przyjmuje rolę lidera.

Ocenę bardzo dobrą otrzymuje uczeń, który:

- 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,
- pisze programy konwertujące liczby między różnymi systemami pozycyjnymi,
- w programach wykonujących działania na liczbach w różnych systemach pozycyjnych wykorzystuje bibliotekę string i strukturalne typy danych,
- wykorzystuje rozwinięcie binarne liczby dziesiętnej w algorytmie szybkiego podnoszenia do potęgi,
- wykonuje operacje arytmetyczne na liczbach w różnych systemach, implementuje je w języku C++,

Ocenę dobrą otrzymuje uczeń, który:

- pisze programy konwertujące liczby między systemem dziesiętnym i binarnym,
- implementuje w języku C++ algorytmy wykonujące działania arytmetyczne na liczbach w różnych systemach,
- w algorytmach zamiany wykorzystuje zależności między systemami binarnym, ósemkowym i heksadecymalnym,
- omawia sposób reprezentacji obrazów w komputerze, korzystając z takich pojęć jak: piksel, model RGB, kanał alfa,
- wyjaśnia, na czym polega digitalizacja (dyskretyzacja) dźwięku,
- wyjaśnia zasadę tworzenia animacji,

Ocenę dostateczną otrzymuje uczeń, który:

- dodaje liczby binarne,
- konwertuje liczby między pozycyjnymi systemami liczbowymi,
- wykonuje działania arytmetyczne na liczbach w systemach liczbowych o różnych podstawach,
- przedstawia liczby w kodzie U2,

Ocenę dopuszczającą otrzymuje uczeń, który:

- omawia pojęcia: złożoność obliczeniowa algorytmu, algorytm naiwny, algorytm optymalny, złożoność pesymistyczna, złożoność oczekiwana (średnia),
- definiuje pojęcie systemów liczbowych,
- wymienia systemy liczbowe używane w informatyce,
- konwertuje liczby między systemami binarnym i decymalnym,
- dodaje pisemnie liczby binarne,

Rozdział 6. Relacyjne bazy danych

Ocenę celującą otrzymuje uczeń, który:

- projektuje rozbudowane relacyjne bazy danych, zarządza nimi, wykorzystując zaawansowane narzędzia oraz klauzule języka SQL,
- tworzy rozwinięte responsywne aplikacje internetowe wymagające dużego nakładu pracy i znajomości nowoczesnych technologii,

Ocenę bardzo dobrą otrzymuje uczeń, który:

- projektuje zaawansowane relacyjne bazy danych, zarządza nimi, tworzy tabele pomostowe, formularze, kwerendy i raporty,
- formułuje zapytania w języku SQL, stosując selekcję, sortowanie, projekcję oraz agregowanie danych,
- tworzy aplikacje internetowe z przejrzystym interfejsem użytkownika korzystające z sieciowej bazy danych, testuje je i wprowadza poprawki,
- wyszukuje informacje w bazach danych, stosując różne techniki (w tym konstruowanie rozbudowanych
- zapytań),
- umie wprowadzić dane bezpośrednio do tabeli lub poprzez formularz,
- umie poprawniw korzystać z formularzy w celu wprowadzania, wyszukiwania i edycji danych,
- potrafi drukować tabele i formularze,
- tworzy i modyfikuje formularze oraz formanty za pomocą kreatora i w widoku projektu,
- dodaje etykiety, formanty, nagłówek lub stopkę do formularza, zmienia jego szatę graficzną,

- tworzy proste raporty przy użyciu narzędzia Autoraport,
- umie korzystać z kreatora raportów do generowania dowolnych rodzajów raportów,
- potrafi grupować informacje w raporcie,
- umie korzystać z podsumowania statystycznego,
- przygotowuje raport do druku,
- definiuje makropolecenia składające się z sekwencji kilku akcji,
- stosuje podstawowe mechanizmy chroniące plik bazy danych przed jego otwarciem i ewentualną przypadkową lub celową modyfikacją,
- rozumie konieczność wykonywania kopii bezpieczeństwa (na przykład podczas wprowadzania i testowania kwerend funkcjonalnych).
- zna sposób i rozumie celowość kodowania bazy danych,

Ocenę dobrą otrzymuje uczeń, który:

- projektuje i tworzy proste bazy danych,
- korzysta z danych przechowywanych w innych bazach programu Access oraz utworzonych w innych aplikacjach,
- samodzielnie projektuje proste zapytania, korzystając z widoku projektu i kreatorów różnego typu kwerend,
- stosuje odpowiednie kryteria i parametry do wyszukiwania danych,
- definiuje złożone kryteria wyboru danych,
- konstruuje pola obliczeniowe, wprowadza korekty,
- stosuje kwerendy parametryczne,
- tworzy kwerendy funkcjonalne generujące tabele, aktualizujące dane, usuwające i dołączające dane,
- tworzy zestawienia krzyżowe, odpowiednio definiując kryteria, nagłówki wierszy i kolumn,

Ocenę dostateczną otrzymuje uczeń, który:

- wyszukuje informacje w bazach danych, tworzy formularze, kwerendy i raporty,
- wykorzystuje język SQL do tworzenia i usuwania baz danych, dodawania tabel do baz danych, usuwania tabel z baz, dodawania rekordów do tabel, importowania danych do tabel, edycji rekordów,
- tworzy konta użytkowników i przydziela im uprawnienia do wybranej bazy, używając języka SQL,
- formułuje zapytania zwracające określone dane, sortuje wyniki zapytań,
- potrafi zaprojektować układ tabel, unikając powtarzania danych,
- potrafi zaimportować tabele z istniejącej bazy danych, arkusza kalkulacyjnego czy pliku tekstowego,
- tworzy tabele przy użyciu kreatora tabel oraz w widoku projektu,
- potrafi ustalić właściwości pól,
- zna różne formaty danych,

- zna pojęcia: klucz główny oraz klucz obcy,

Zna i rozumie pojęcia: system zarządzania danymi, relacyjna baza danych.
Zna zasady projektowania bazy danych.
Zna właściwości rekordów i pól bazy danych i rozumie różnice pomiędzy nimi.

- dostrzega korzyści wynikające ze stosowania kluczy głównych,
- łączy tabele odpowiednimi relacjami,
- rozróżnia podstawowe typy relacji między tabelami, potrafi je nazwać,
- projektuje tabele, uwzględniając różne typy danych,
- projektuje relacyjne bazy danych z uwzględnieniem zjawisk redundancji. Zapewnia integralność danych,
- wykonuje podstawowe czynności edycyjne na obiektach bazy danych (kopiowanie, usuwanie, zmiana nazwy),
- potrafi wstawiać obiekty OLE,
- odróżnia osadzanie obiektu w tabeli od jego połączenia,
- potrafi zmieniać wygląd tabeli w widoku arkusza danych,
- zmienia kolejność sortowania danych w tabeli,
- potrafi porządkować tabele i przeglądać wybrane rekordy przez sortowanie jedno- i wieloparametrowe,
- stosuje filtry do wyszukiwania informacji,

Ocenę dopuszczającą otrzymuje uczeń, który:

- wie, co to jest język SQL, zna podstawowe klauzule tego języka,
- zna zasady tworzenia zapytań do bazy z wykorzystaniem języka SQL,
- na przykładzie istniejącej relacyjnej bazy danych wymienia obiekty związane z takimi bazami (tabele, kwerendy, formularze, raporty, makrodefinicje).

* opracowano na podstawie materiałów wydawnictwa Nowa Era – Informatyka na czasie.