Wymagania – Informatyka klasa 2 Poziom rozszerzony

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, schemat blokowy, 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	– wykonuje zadania o podwyższonym stopniu trudności: oznaczone trzema gwiazdkami w podręczniku, z arkuszy maturalnych z lat poprzednich lub konkursów i olimpiad informatycznych

			– omawia działanie operacji logicznych i reprezentację ich wyników w komputerze (wynik może przyjmować wartość prawda – 1 lub fałsz – 0, co zajmuje 1 bajt pamięci) – opisuje istotę cyfrowej reprezentacji w komputerze obrazów, dźwięków i animacji	
3	Algorytmy zamiany reprezentacji liczb między systemami liczbowymi	3	– tworzy programy do konwersji między liczbami w systemach binarnym i decymalnym – 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 decymalnego 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 – wyznacza liczby bliźniacze	– implementuje algorytmy dotyczące liczb pierwszych 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 – 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	– 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

			— stosuje w swoich programach operacje plikowe – wczytywanie danych z pliku dyskowego, zapis wyniku do pliku	— tworzy własne funkcje, dobierając sposób przekazywania parametrów
9	Łamiemy szyfr Cezara	3	— wyjaśnia, na czym polega łamanie szyfru (kryptoanaliza) — 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 wypisujący jego pozycję (algorytm „dziel i zwyciężaj”)	— opisuje różne sposoby łamania szyfrów i implementuje je w języku C++ — pisze programy deszyfrujące o podwyższonym poziomie trudności
10	Poszukujemy liczby	2	— znajduje wartość w zbiorach uporządkowanym i nieuporządkowanym, stosując odpowiednio algorytmy wyszukiwania liniowego, liniowego z wartownikiem i binarnego — pisze programy wykorzystujące przekazywanie parametru 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 wyszukiwania — 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), optymistyczna i pesymistyczna	— określa złożoność czasową i pamięciową algorytmów z zastosowaniem 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 *in situ* 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 sortowania	— 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 wykorzystaniem algorytmów sortowania — ocenia wpływ pierwotnego ułożenia danych w zbiorze na liczbę wykonywanych operacji

13	Szyfry przestawieniowe, anagramy	2	– omawia zasadę działania szyfrów przestawieniowych, wymienia przykłady takich szyfrów – sprawdza, czy słowa (napisy) są anagramami – pisze funkcje sprawdzające – wykorzystuje poznane wcześniej algorytmy sortowania i zliczania w rozwiązywaniu problemów	– pisze programy szyfrujące o podwyższonym stopniu trudności – wyszukuje anagramy w plikach tekstowych – posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym – do rozwiązania problemu dobiera optymalny algorytm i struktury danych
14	Sito Eratostenesa	2	– opisuje algorytmy sprawdzające, czy liczba jest pierwsza – omawia i stosuje algorytm sita Eratostenesa do wyszukiwania liczb pierwszych w określonym przedziale liczbowym – określa złożoność obliczeniową algorytmu	– pisze programy o podwyższonym stopniu trudności wykorzystujące sito Eratostenesa – posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym – optymalizuje algorytm, dążąc do minimalnej złożoności obliczeniowej
15	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 wyszukujące spójne podciągi – posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym – do rozwiązania problemu dobiera optymalny algorytm i struktury danych
16	W poszukiwaniu lidera i idola	2	– definiuje pojęcia idola w grupie i lidera w zbiorze – znajduje idola w grupie lub stwierdza jego brak – określa, czy w zbiorze jest lider – omawia i implementuje w języku C++ algorytmy szukania idola oraz lidera – ocenia złożoność obliczeniową stosowanych algorytmów i ich efektywność – stosuje tablice dwuwymiarowe w pisanych programach – stosuje funkcję sort z biblioteki STL do wyszukiwania lidera	– pisze programy o podwyższonym stopniu trudności – posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym – do rozwiązania problemu dobiera optymalny algorytm i struktury danych

17	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
18	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 telewidza/kinomana (optymalny harmonogram wykorzystania sali), pakowanie plecaka, 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
19	Programowanie dynamiczne	5	– wyjaśnia, na czym polega metoda dynamiczna – implementuje optymalne algorytmy dotyczące problemu kasjera, telewidza, znajdowania drogi oraz pakowania plecaka – stosuje metodę dynamiczną do znajdowania najdłuższego wspólnego podciągu – porównuje metody zachłanną i dynamiczną	– pisze programy o podwyższonym stopniu trudności wykorzystujące algorytmy dynamiczne – posługuje się środowiskiem programistycznym oraz językiem programowania w stopniu zaawansowanym – do rozwiązania problemu dobiera optymalny algorytm i struktury danych
20	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