Wymagania edukacyjne z informatyki klasa 1 – zakres rozszerzony

Temat	Ocena dopuszczająca	Ocena dostateczna	Ocena dobra	Ocena bardzo dobra	Ocena celująca
I. Przestrzeganie prawa i zasad bezpieczeństwa podczas pracy przy komputerze					
Bądź uczciwy, czyli przestrzeganie prawa w świecie informatyki	– wie, że udostępnianie treści chronionych prawem autorskim jest przestępstwem – podaje przykłady łamania praw autorskich – szanuje własność intelektualną	– wie, kiedy nie narusza prawa podczas korzystania z utworów z sieci – wie, czym charakteryzuje się licencja CC i na jakich zasadach można używać takich dzieł – określa, czym w świetle prawa jest utwór	– wskazuje legalne źródła muzyki, grafiki, animacji itp. rozpowszechnianej na licencji CC – wskazuje różnice pomiędzy plagiatem a cytatem	– określa i przedstawia zasady legalnego korzystania z dzieł objętych prawami autorskimi	– omawia niektóre metody działania instytucji i kancelarii prawnych w zakresie ścigania osób łamiących prawo autorskie
8, 16, 32, 64, czyli jak rozwój technologii wpływa na rozwój społeczeństw	– wie, że istnieją inne systemy liczbowe poza dziesiętnym i tłumaczy ich zastosowanie – zna pojęcia *bajt* i *bit* – wie, jak powstają wagi poszczególnych pozycji w kodzie binarnym – wie, jaki wpływ na zastosowanie komputerów ma postęp technologiczny	– umie wykorzystać kalkulator do prezentacji liczb w różnych systemach liczbowych – omawia zalety zdalnego nauczania i jego wpływu na rozwój społeczny – podaje przykłady wpływu postępu technologicznego na rozwój informatyki	– wie, dlaczego do projektowania układów komputera używa się kodu dwójkowego – omawia zalety zdalnego nauczania i jego wpływu na rozwój społeczny – omawia wpływ rozwoju technologii informacyjnych na rozwój społeczeństw	– omawia zmiany technologiczne poszczególnych elementów komputerów i ich wpływ na zastosowanie komputerów	– analizuje tendencje rozwoju społeczeństwa w kontekście rozwoju technologicznego
Chrońmy informacje, czyli techniki uwierzytelniania, kryptografia i podpis elektroniczny	- wie czym są techniki uwierzytelniania, - wie czym jest kryptografia - wie czym jest i gdzie jest	- zna techniki uwierzytelniania - zna kody jedno- i dwuczęściowe - zna kody jednorazowe	- umie rozróżnić techniki uwierzytelnienia w zależności od tego, co jest uwierzytelnianie - omawia kody jedno- i	- zna i stosuje zasady uwierzytelniania dwuskładnikowego, - określa techniki kryptografii	- stosuje łącznie dwie różne techniki uwierzytelniania

	wykorzystywany podpis elektroniczny	- zna superszyfrowanie - zna podpis cyfrowy	dwuczęściowe, - omawia kody jednorazowe - omawia superszyfrowanie - omawia podpis cyfrowy	- zna i omawia kodowanie i szyfrowanie - omawia procedury prawne podpisu elektronicznego - stosuje podpis elektroniczny w życiu codziennym	
Kim jestem, czyli jak bezpiecznie budować wizerunek w sieci	– określa, czym są przepisy oparte na RODO i jaki jest cel ich wprowadzenia	– określa, na czym polegają prawa obywatela do ochrony wizerunku i wskazuje źródła prawa – określa, czym grozi upowszechnianie wizerunku bez zgody danej osoby – wie, co zrobić w przypadku wykrycia naruszenia swoich praw do wizerunku	– umie opisać cyberzagrożenia i wskazać najgroźniejsze z punktu widzenia przepisów o ochronie wizerunku – omawia zasady bezpiecznego korzystania z sieci i usług sieciowych w kontekście ochrony własnego wizerunku i niewykorzystywania cudzego bez odpowiedniej zgody	– wymienia działania, które służą ochronie wizerunku w sieci, i wie, jak nie naruszać tych praw – opracowuje własne zasady ochrony wizerunku na podstawie przepisów prawa – wie, czym może skutkować kradzież tożsamości	– określa i objaśnia kolegom na lekcji zagrożenia płynące z możliwości kradzieży tożsamości w kontekście oszustw i wyłudzeń
II. Podstawy programowania i środowisko programistyczne					
Przypomnij sobie, czyli podstawy tworzenia algorytmów	– wie, ze algorytm można zapisać w różnych postaciach i wymienia nazwy tych sposobów – zna podstawowe zasady tworzenia schematów blokowych w tym dozwolone i niedozwolone połączenia	– prawidłowo interpretuje działanie bloku decyzyjnego i wie, jaką rolę odgrywa on w algorytmie – wie, jaka jest różnica pomiędzy blokiem decyzyjnym a wejściowym/wyjściowym	– samodzielnie analizuje przykład algorytmu, np. z podręcznika z pomocą opisu – buduje algorytmy prostych zagadnień z różnych dziedzin lub przedmiotów szkolnych – umie ułożyć prosty algorytm w postaci schematu blokowego na podstawie algorytmu typu	– samodzielnie tworzy algorytmy na podstawie specyfikacji zawierające bloki decyzyjne – samodzielnie testuje algorytmy dla różnych przypadków – samodzielnie dyskutuje sposób rozwiązania problemu na podstawie algorytmu, np. z podręcznika, i proponuje	– analizuje różne algorytmy i testuje je na samodzielnie i odpowiednio dobranych danych – samodzielnie proponuje modyfikacje przykładowych algorytmów

			lista kroków zawierający blok decyzyjny	jego modyfikacje	
Algorytm, czyli utrwalamy swoją wiedzę	- wie czym jest schemat blokowy - wie co to jest pseudokod - wie czym jest algorytm zapisany w formie listy kroków	- umie zapisać algorytm wykorzystując schemat blokowy - umie zapisać algorytm wykorzystując listę kroków - umie zapisać algorytm przy pomocy pseudokodu	- stosuje tworzenie algorytmów w postaci schematu blokowego - stosuje tworzenie algorytmów w postaci listy kroków - stosuje tworzenie algorytmów w postaci pseudokodu	- samodzielnie analizuje złożone algorytmy zapisane w dowolnej postaci	- samodzielnie tworzy, analizuje i testuje algorytmy dla bardziej złożonych problemów
Pozycyjne reprezentacje liczb, czyli tablice w algorytmie	- wie co to są liczby doskonałe - wie czym są tablice - wie czym jest schemat Hornera	- opisuje czym jest doskonałość - umie nazwać czym są liczby doskonałe - opisuje czym są tablice	- umie tworzyć algorytmy z wykorzystaniem tablic - umie tworzyć algorytmy z wykorzystaniem liczb doskonałych, parzystych i pierwszych	- umie tworzyć algorytmy wykorzystujące schemat Hornera - - umie tworzyć algorytmy z wykorzystaniem pozycyjnych reprezentacji liczb	- samodzielnie tworzy, analizuje i testuje algorytmy dla bardziej złożonych problemów wykorzystujących schemat Hornera oraz liczby doskonałe
Narzędzia, czyli jak dobrać i skonfigurować środowisko programistyczne	– wymienia nazwy kilku języków programowania – wie, gdzie sprawdzić poziom popularności niektórych z nich – wie, co powinno wpływać na dobór odpowiedniego języka programowania	– zna podstawowe cechy języków JAVA i C++ – wie, czym jest środowisko programistyczne – umie edytować kod programu w Eclipse	– umie instalować Eclipse i przystosować środowisko do danego języka programowania	– umie uruchamiać w Eclipse program ułożony w JAVA – umie uruchomić polski pakiet językowy dla Eclipse	– umie omówić działanie poszczególnych linii przykładowego kodu programu w JAVA
C z plusami, czyli uruchamiamy pierwsze programy w C++	– wie, że Eclipse należy przystosować do układania programów w C++ poprzez zainstalowanie pakietu MinGW	– z pomocą nauczyciela instaluje Eclipse dla C++ i pakiet MinGW Na podstawie podręcznika: – umie utworzyć nowy projekt w C++ w IDE	Na podstawie podręcznika: – omawia działanie i znaczenie poszczególnych linii programu testowego Hello World – sprawnie posługuje się środowiskiem Eclipse w	– samodzielnie instaluje i uruchamia środowisko programistyczne Eclipse dla C++ – samodzielnie uruchamia program testowy i umie go modyfikować	– umie instalować inne środowiska programistyczne dla C++ oraz edytować i uruchamiać w nich programy

		Eclipse – uruchamia program testowy Hello World	zakresie edycji, uruchamiania i zachowywania programów w C++		
C z plusem, czyli podstawy programowania	– wymienia i charakteryzuje kolejne etapy tworzenia programu komputerowego – wie, czym jest badanie warunku w programie i kiedy się je stosuje w kontekście bloków warunkowych algorytmu – wie, że istnieją różne typy operatorów i na podstawie podręcznika omawia rolę niektórych z nich – wie, że w programie mogą występować biblioteki i funkcje	– korzystając z podręcznika, omawia podstawową strukturę programu w języku C++ – zna znaczenie nawiasów klamrowych i "//" oraz ich rolę w programie C++ – na podstawie tabeli z podręcznika omawia znaczenie operatorów – odróżnia operatory arytmetyczne od relacyjnych i logicznych i symboli porównawczych – wie, czym są zmienne i stałe w programie komputerowym i wskazuje ich deklarację w przykładowym programie – zna znaczenie i rolę funkcji i bibliotek – zna pojęcie pętli i warunku	– samodzielnie omawia działanie poszczególnych operatorów wszystkich typów – umie zapisać warunki dla instrukcji warunkowej – umie dołączyć bibliotekę do kodu programu – wymienia różnice pomiędzy instrukcją warunkową a pętlą	– samodzielnie układa proste programy ze strumieniowaniem – samodzielnie charakteryzuje najczęściej używane typy zmiennych liczbowych i logicznych – samodzielnie zapisuje fragment programu z instrukcją warunkową na podstawie algorytmu – wskazuje, w których przypadkach należy użyć danej instrukcji warunkowej lub pętli	– samodzielnie stosuje instrukcje warunkowe i pętle w prostych programach układanych na podstawie algorytmów
C z plusem, czyli podstaw programowania ciąg dalszy	– omawia na zasadzie analogii budowę zmiennych tablicowych 1 lub 2-wymiarowych – umie graficznie przedstawić tablice	– omawia różnice pomiędzy zmiennymi a zmiennymi tablicowymi – podaje przykłady zastosowania indeksowania pozycji, np. w sali kinowej	– w programie komputerowym w C++ umie zastosować strumieniowanie danych do wprowadzenia wartości danych lub wyprowadzenia na ekran	– samodzielnie układa proste programy, w których używa strumieniowania danych i instrukcji warunkowych oraz pętli – samodzielnie opracowuje	– proponuje rozwiązania problemów informatycznych wskazanych przez nauczyciela zawierających instrukcje warunkowe, pętle i

		–wie, że do wyprowadzania i wprowadzania informacji w programie komputerowym stosuje się strumieniowanie	tekstów lub wartości zmiennych – korzystając z podręcznika, umie analizować programy, w których zastosowano strumieniowanie danych – z pomocą podręcznika stosuje w prostych programach pętle i instrukcje warunkowe, np. do wypełniania tablicy danymi	sposób wyświetlania i wprowadzania danych – umie stosować instrukcje zagnieżdżone	strumieniowanie
III. Programowanie i algorytmy					
Wspólna miara, czyli podstawy złożoności obliczeniowej	- wie na czym polega złożoność obliczeniowa - wie czym jest notacja omikron - wie czym jest operacja dominująca	- umie objaśnić czym jest złożoność obliczeniowa - objaśnia operację dominującą - objaśnia złożoność pamięciową - omawia czym jest złożoność optymistyczna	- omawia czym jest złożoność optymistyczna - omawia notację omikron	– samodzielnie stosuje w praktyce czasową złożoność obliczeniową – samodzielnie tworzy algorytm wykorzystujący znak klasy złożoności	– samodzielnie układa algorytm dla bardziej złożonych problemów wykorzystujący czasową złożoność obliczeniową
Funkcje – piszemy własne podprogramy Funkcje –zmienne lokalne i globalne, przesłanianie zmiennych Funkcje – przekazywanie parametrów	– zna budowę funkcji; – wie, na czym polega wywołanie funkcji; – zna różnicę pomiędzy działaniem i zastosowaniem funkcji zwracającej wartość i funkcji void; – określa przypadki, w których niezbędne jest zastosowanie funkcji zwracającej wartość i funkcji void;	– wywołuje funkcje w treści programu; – analizuje działanie poszczególnych funkcji;	– stosuje unikalne nazwy zmiennych wewnątrz bloku; – prawidłowo układa mało skomplikowane funkcje wykorzystujące globalne i lokalne zmienne; – wywołuje funkcje z parametrami; – prawidłowo i czytelnie dobiera nazwy zmiennych w deklaracjach funkcji;	– omawia istotę deklaracji wewnętrznych zmiennych i stałych, typów i podprogramów lokalnych; – układa i wywołuje funkcje z parametrami i bez; – prawidłowo układa funkcje zwracające wartość i void wykorzystujące globalne i lokalne zmienne;	– samodzielnie planuje użycie funkcji do rozwiązywania problemów informatycznych; – samodzielnie układa funkcje, stosując zmienne lokalne i globalne;

			– wie, jak działa przekazywanie przez wartość; – wie, jak działa przekazywanie przez wskaźnik;	– stosuje unikalne nazwy zmiennych wewnątrz bloku; – układa funkcje z przekazywaniem parametrów przez wartość lub wskaźnik;	
Przez 1 i siebie, czyli jak zbadać, czy liczba jest liczbą pierwszą	– zna definicję liczby pierwszej i umie wymienić kilka z nich, wskazując spełnienie podstawowej cechy – wymienia nazwę metod badania, czy liczba jest liczbą pierwszą	– na podstawie tabeli z podręcznika umie objaśnić metodę sita Erastotenesa – objaśnia algorytmy badania, czy liczba jest liczbą pierwszą – zna zastosowanie liczb pierwszych	– zna algorytmy zapisane w różnych postaciach wykrywające liczby pierwsze – sprawdza na przykładach działanie algorytmów wykrywających liczby pierwsze	– samodzielnie tworzy i omawia działanie algorytmów wykrywających liczby pierwsze – przedstawia algorytmy w różnych zapisach w tym schematu blokowego i listy kroków – samodzielnie testuje algorytmy i dobiera odpowiednie dane wejściowe	– samodzielnie układa algorytm testowania liczb na podstawie opisu metody
Która pierwsza, czyli jak program bada liczby	– rozumie działanie algorytmów przedstawionych w podręczniku	– na podstawie algorytmów z podręcznika omawia metodę badania liczby opartą na obliczaniu reszty z dzielenia – wskazuje fragmenty przykładowego programu zgodne z poszczególnymi fragmentami algorytmu	– analizuje programy z podręcznika, omawiając znaczenie poszczególnych instrukcji – testuje programy i algorytmy na przykładowych danych – omawia modyfikacje algorytmu dla większej liczby danych	– samodzielnie układa algorytmy i modyfikuje je wg wskazań nauczyciela – samodzielnie układa program na podstawie algorytmu – samodzielnie testuje program i poprawia ewentualne błędy, omawiając przyczynę ich powstania	– samodzielnie układa program badający liczby wpisane do tablic o 1 lub więcej wymiarach
Więcej pierwszych, czyli generujemy liczby	- wie do czego w informatyce wykorzystuje się liczby pierwsze	- umie wygenerować liczbę pierwszą z niedużych liczb naturalnych za pomocą algorytmu	- analizuje algorytm, który wyłania liczby pierwsze z zadanego zakresu liczb	– samodzielnie układa i modyfikuje je wg wskazań nauczyciela, który wyłania liczby pierwsze z zadanego zakresu liczb	- samodzielnie układa i modyfikuje program który wyłania liczby pierwsze z zadanego zakresu liczb

Największy i najmniejszy, czyli jak znaleźć NWD i NWW	– definiuje NWD i omawia jego zastosowanie w matematyce – podaje kilka przykładów NWD dla wybranych liczb	– na podstawie gotowego zapisu przykładu algorytmu Euklidesa, np. z podręcznika, omawia istotę tej metody – podaje i uzasadnia dziedzinę liczb, dla których przeznaczony jest algorytm Euklidesa	– omawia różnicę pomiędzy metodą rekurencyjną a iteracyjną – analizuje gotowy przykład zastosowania metod Euklidesa – przedstawia algorytmy Euklidesa, np. w formie schematu blokowego, i tłumaczy ich istotę	– analizuje obie metody Euklidesa pod kątem wydajności i szybkości działania dla różnych zestawów zmiennych wejściowych	– samodzielnie przeprowadza analizę wydajności algorytmu Euklidesa dla różnych danych i przewiduje wyniki swojej analizy
Euklides i rekurencja, czyli szukamy NWD	- wie czym jest największy wspólny dzielnik (NWD) - wie czym jest rekurencja	- umie wyszukać NWD - układa algorytm umożliwiający wyszukanie NWD - umie zastosować w praktyce rekurencję	- omawia ciąg Fibonacciego - układa algorytmy i tłumaczy ich działanie z zastosowaniem ciągu Fibonnaciego	- omawia algorytm Euklidesa - umie zastosować w praktyce algorytm Euklidesa układając algorytm rekurencyjny NWD	– samodzielnie układa algorytm i pisze program na jego podstawie dla bardziej złożonych problemów wykorzystujący algorytm Euklidesa
Metoda iteracji, czyli algorytm Euklidesa w innej postaci	- wie czym jest iteracja	- omawia algorytm iteracyjny	- umie zastosować pętle w praktyce - zna instrukcje pętli w językach C++ i Java	- umie zastosować iteracje w algorytmie Eukidesa	– samodzielnie układa algorytm i pisze program na jego podstawie dla bardziej złożonych problemów wykorzystujący iterację w algorytmie Euklidesa
Dodawanie ułamków, czyli jak wykorzystać NWW i NWD w programie komputerowym	– zapisuje algorytm dodawania ułamków w postaci listy kroków lub schematu blokowego	– analizuje program ułożony na podstawie algorytmu dodawania ułamków	– analizuje przykładowy program z podręcznika z użyciem funkcji i omawia ich rolę	– samodzielnie układa program dodający ułamki i organizuje sposób wprowadzania danych i wyświetlania wyników	– biegle posługuje się samodzielnie ułożonymi funkcjami – obliczającymi wyniki pośrednie, a także organizującymi wydruk i wprowadzanie danych
Skracamy i wyłączamy, czyli dodawania ułamków ciąg dalszy	– zapisuje algorytm upraszczania ułamków na podstawie ogólnej wiedzy matematycznej	– określa rolę NWW i NWD w algorytmie upraszczania ułamków – omawia algorytm wyłączania całości z	– analizuje program upraszczający ułamki i wskazuje w nim rolę funkcji obliczających NWW i NWD	– samodzielnie układa program wykorzystujący funkcje NWW i NWD do upraszczania ułamków – samodzielnie układa	– samodzielnie układa program dodający ułamki

		ułamka	– proponuje różne rozwiązania problemu wyświetlania wyników – analizuje program wyłączania całości z ułamka	program wyłączający całości z ułamka	
Szyfrowanie, czyli poznajemy szyfr Cezara i szyfr przedstawieniowy	– omawia cele szyfrowania danych i informacji – tłumaczy, na czym polega podstawieniowy sposób szyfrowania informacji – wie, jak odróżnić strony internetowe z szyfrowaną transmisją danych od pozostałych	– na przykładzie tabeli tłumaczy metodę przestawieniową i umie zaszyfrować tekst tą metodą – omawia na podstawie rysunku z podręcznika metodę szyfrowania szyfrem Cezara	– wie, na czym polega szyfrowanie szyfrem wieloalfabetowym – tłumaczy potrzebę szyfrowania niektórych transmisji w sieci	– samodzielnie układa algorytm dla szyfru Cezara	– samodzielnie układa program komputerowy szyfrujący szyfrem Cezara
Konspiracja, czyli tworzymy własne szyfry	– wie, jaką rolę odgrywa klucz w metodzie szyfrowania	– odróżnia metodę szyfrowania symetryczną od niesymetrycznej – na przykładzie przedstawia działanie klucza – wie, jak generować liczbę pseudolosową	– prawidłowo, na podstawie podręcznika, analizuje działanie programu szyfrującego stałym kluczem – tłumaczy na przykładzie z podręcznika rolę tablicy stałej kluczy – objaśnia metodę stosowania losowo generowanych kluczy raz ich wady i zalety	– samodzielnie układa programy szyfrujące z kluczem zapisanym w tablicy – samodzielnie układa program oparty na losowo generowanych danych w tablicach kluczy	– samodzielnie modyfikuje program szyfrujący losowo generowanym kluczem w taki sposób, by szyfrował także klucze przesyłane odbiorcy informacji
Z kluczem publicznym, czyli bezpieczniejsze szyfrowanie	- wie czym jest szyfrowanie kluczem publicznym	- omawia czym jest szyfr asymetryczny (RSA) - wie, ze szyfrowanie asymetryczne działa w trzech etapach	- zna etapy działania algorytmu RSA	- umie stworzyć algorytm szyfr wykorzystujący algorytm RSA	- samodzielnie pisze program na podstawie algorytmu szyfru wykorzystującego algorytm RSA - samodzielnie pisze program deszyfrujący wykorzystujący algorytm RSA

Szyfrowanie zachłanne, czyli Kody Huffmana	- wie czym jest metoda zachłanna i w jakim celu ją się stosuje	- omawia zastosowanie metody zachłannej w szyfrowaniu - omawia metodę optymalną	- stosuje metodę optymalną w tworzeniu algorytmów - omawia zastosowanie Kodów Huffmana	- tworzy algorytmy wykorzystujące Kody Huffmana	- stosuje Kody Huffmana podczas kodowania znaków ASCII
Automat wydaje monety, czyli jak dynamicznie wydawać resztę	– wie, na czym polega metoda zachłanna wydawania reszty – tłumaczy działanie algorytmu z podręcznika	– samodzielnie układa algorytm wydawania reszty metodą zachłanną dla nominałów większych lub równych 1 zł – testuje programy i algorytmy z podręcznika na przykładowych zmiennych	– samodzielnie układa algorytm wydawania reszty dla wszystkich możliwych nominałów – objaśnia działanie programu wydającego resztę w kontekście algorytmu podanego w podręczniku	– samodzielnie układa algorytm i program wydający resztę metodą zachłanną korzystający ze wszystkich dostępnych nominałów – omawia problemy, które należało rozwiązać podczas układania algorytmu i programu dla dowolnych nominałów	– modyfikuje program dla innego układu monetarnego, np. z USA
Wydajemy resztę, czyli jak wykorzystać metodę zachłanną	- wie czym jest metoda zachłanna	- objaśnia zasadę wydawania reszty przez automat z wykorzystaniem różnych nominałów	- układa algorytm zachłanny do stworzenia algorytmu rozwiązującego problem wydawania reszty w różnych nominałach przez automat	- pisze program z wykorzystaniem metody zachłannej	- układa program wykorzystujący algorytm zachłanny do stworzenia programu rozwiązującego problem wydawania reszty w różnych nominałach przez automat
Co z tym groszem, czyli błąd zaokrąglenia i błąd przybliżenia	- wie w jak zaokrągla się liczby - wie w jakim celu zaokrągla się liczby	- omawia błąd zaokrąglenia - omawia błędy względne i bezwzględne w zaokrągleniu	- wie dlaczego stosujemy zaokrąglanie liczb z nadmiarem i niedomiarem	- umie wykorzystać zaokrąglanie liczb z nadmiarem i niedomiarem - umie wykorzystać problemy błędu zaokrąglania, błędu bezwzględnego i błędu względnego do rozwiązania zadań	- układa samodzielnie i modyfikuje programy, które rozwiązują problem zaokrąglania oraz błędy występujące przy zaokrąglaniu
Sprawdzam, czyli porównywanie tekstów i	- wie czym jest wzorzec - wie czym jest	- umie zastosować wzorzec do rozwiązania	- umie stworzyć algorytm naiwny w celu	- stosuje implementację podczas wyszukiwania	- umie stworzyć program i go zmodyfikować

wyszukiwanie wzorca w tekście metodą naiwną	implementacja podczas szukania wzorca	zadanego problemu - omawia czym jest metoda naiwna w algorytmice	wyszukiwania wzorca w łańcuchu tekstowym	wzorca w tekście - umie stworzyć algorytm wykorzystując implementacje podczas wyszukiwania wzorca	wykorzystujący implementację podczas wyszukiwania wzorca
Od ostatniego, czyli szukamy wzorca tekstu metodą Boyera-Moore'a	- zna algorytm Boyera-Moore'a - wie czym jest czym jest podliniowa klasa złożoności	- umie zastosować wyszukiwanie wzorca algorytmem BM	- umie wykorzystać wzorzec BM do wyszukiwania w tekście z użyciem tablicy	- umie wykorzystać uproszczony algorytm BM do wyszukiwania wzorca	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm BM do wyszukiwania wzorca
Z lewej i z prawej, czyli szukamy palindromów	- wie czym jest palindrom - wie co to jest algorytm naiwny	- omawia zastosowanie palindromu w informatyce	- umie stworzyć algorytm naiwny wyszukujący palindromy	- umie zastosować algorytm naiwny do sprawdzenia, czy dany ciąg znaków jest palindromem	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm naiwny do sprawdzenia, czy dany ciąg znaków jest palindromem
Przestawianie liter, czyli jak stworzyć anagram	- wie czym jest anagram	- wie jakie zastosowanie ma anagram w informatyce	- umie stworzyć algorytm sortowania bąbelkowego	- umie stworzyć algorytm sprawdzający czy tablica jest anagramem z wykorzystaniem sortowania bąbelkowego	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm sprawdzający czy tablica jest anagramem z wykorzystaniem sortowania bąbelkowego
Od A do Z, czyli alfabetyczne sortowanie wyrazów	- wie czym jest sortowanie alfabetyczne - wie czym jest sortowanie jednopoziomowe	- umie zastosować sortowanie alfabetyczne w celu posortowania elementów ciągu znaków - wie czym jest sortowanie przez wybór	- umie zastosować sortowanie jednopoziomowe w celu posortowania ciągu znaków	- umie stworzyć algorytm wykorzystujący sortowanie przez wybór	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm wykorzystujący sortowanie przez wybór
Odwrotna notacja polska, czyli zapisujemy wyrażenia arytmetyczne	- wie czym jest odwrotna notacja polska	- umie zastosować odwrotną notację polską w działaniach arytmetycznych - wie czym jest zapis prefiksowych	- umie zastosować zapis prefiksowy	- umie stworzyć algorytm obliczenia wartości wyrażenia odwrotnej notacji polskiej	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm obliczenia wartości wyrażenia odwrotnej notacji polskiej

Naiwny jak algorytm, czyli szukamy najmniejszego i największego elementu metodą naiwną	- wie czym jest sortowanie naiwne - wie czym jest rekurencja - wie czym jest iteracja	- umie stworzyć algorytm wykorzystujący sortowanie naiwne	- umie stworzyć algorytm sortowania naiwnego w wersji rekurencyjnej	- umie stworzyć algorytm sortowania naiwnego w wersji iteracyjnej	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm sortowania naiwnego w wersji rekurencyjnej - umie stworzyć program i go zmodyfikować wykorzystujący algorytm sortowania naiwnego w wersji iteracyjnej
Dwie dodatkowe tablice, czyli stosujemy algorytm optymalny	- wie czym jest metoda „dziel i rządź” - wie czym jest algorytm optymalny	- umie stworzyć algorytm wykorzystujący metodę „dziel i rządź”	- umie stworzyć algorytm optymalny	- umie stworzyć algorytm jednoczesnego wyszukiwania najmniejszego i największego elementu wykorzystując metodę optymalną	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm jednoczesnego wyszukiwania najmniejszego i największego elementu wykorzystując metodę optymalną
Dziel i zwyciężaj, czyli sortowanie metodą scalania	- wie co to są algorytmy proste - wie co to jest sortowanie metodą scalania	- umie tworzyć i stosować algorytmy proste - umie tworzyć algorytmy wykorzystujące sortowanie metodą scalania	- umie tworzyć i stosować algorytmy wykorzystujące sortowanie rekurencyjne	- wie co to jest scalanie zbiorów uporządkowanych - umie stworzyć algorytm scalania dwóch zbiorów uporządkowanych	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm scalania dwóch zbiorów uporządkowanych
Raz krócej, raz dłużej, czyli sortowanie kubełkowe	- wie czym jest sortowanie kubełkowe	- zna cechy algorytmu sortowania	- umie opracować algorytm sortowania kubełkowego	- umie opracować algorytm sortowania metodą scalania	- umie stworzyć program i go zmodyfikować wykorzystujący algorytm sortowania kubełkowego i sortowania metodą scalania
Powtarzanie w pętli i wywołanie siebie, czyli	– omawia na realnych przykładach różnicę	– na podstawie podręcz-nika lub innych	– samodzielnie analizuje przykładowe algorytmy	– układa algorytmy z podejściem iteracyjnym i	– rozwiązuje problemy obiema metodami

iteracja i rekurencja w algorytmach	między rekurencją a iteracją.	wiarygodnych źródeł analizuje przykładowe algorytmy iteracyjne i rekurencyjne; – na podstawie podręcznika lub innych wiarygodnych źródeł rozpoznaje procesy rekurencyjne i iteracyjne.	iteracyjne i rekurencyjne; – samodzielnie rozpoznaje procesy rekurencyjne i iteracyjne; – wskazuje w przykładowych algorytmach miejsca, które decydują o iteracyjności lub rekurencyjności opisywanego procesu; – analizuje przykładowy program.	rekurencyjnym; – układa programy na podstawie tych algorytmów; – wskazuje instrukcje, które decydują o iteracyjności lub rekurencyjności podejścia do realizacji algorytmu.	i ocenia ich skuteczność.
Sortowanie bąbelkowe, czyli każda liczba jest mniejsza od maksymalnej lub jej równa	– na podstawie podręcznika lub innych wiarygodnych źródeł omawia budowę tablicy jednowymiarowej; – na podstawie podręcznika lub innych wiarygodnych źródeł omawia istotę sortowania bąbelkowego.	– na podstawie podręcznika lub innych wiarygodnych źródeł analizuje działanie algorytmu sortowania bąbelkowego w postaci listy kroków i schematu blokowego; – na podstawie podręcznika lub innych wiarygodnych źródeł analizuje przykład sprawdzający poprawność działania algorytmu.	– samodzielnie omawia istotę metody sortowania bąbelkowego; – samodzielnie omawia działanie przykładowego algorytmu opartego o metodę sortowania bąbelkowego; – samodzielnie sprawdza działanie algorytmu na przykładach.	– układa algorytm sortowania bąbelkowego; – weryfikuje poprawność działania programu na przykładach.	– układa program sortujący metodą bąbelkową w innym języku niż C++ (np. Java).
Przez wstawianie, czyli jeszcze o porządkowaniu liczb	– na podstawie podręcznika lub innych wiarygodnych źródeł omawia istotę sortowania przez wstawianie.	– na podstawie podręcznika lub innych wiarygodnych źródeł analizuje działanie algorytmu sortowania bąbelkowego w postaci listy kroków i schematu przez wstawianie; – na podstawie podręcz-	– samodzielnie omawiana przykładzie istotę metody sortowania przez wstawianie; – samodzielnie omawia działanie przykładowego algorytmu opartego o metodę sortowania przez wstawianie; – samodzielnie sprawdza	– układa algorytm sortowania przez wstawianie w postaci listy kroków i schematu blokowego; – weryfikuje poprawność działania programu na przykładach.	– układa program sortujący metodą przez wstawianie w innym języku niż C++ (np. Java).

		nika lub innych wiarygod- nych źródeł analizuje przykład sprawdzający poprawność działania algorytmu.	działanie algorytmu na przykładach.		
Komputer porządkuje, czyli układamy programy sortujące	– na podstawie podręcz- nika lub innych wiarygod- nych źródeł weryfikuje działanie przykładowych programów (np. z pod- ręcznika).	– na podstawie podręcz- nika lub innych wiarygod- nych źródeł analizuje przykładowe programy i wskazuje miejsca, w których są wykonywane działania najważniejsze dla metody.	– na podstawie podręcz- nika lub innych wiarygod- nych źródeł układa programy sortujące; – na podstawie podręcz- nika uzasadnia użycie danych instrukcji w przykładowych programach.	– samodzielnie układa programy sortujące według obu metod; – samodzielnie weryfikuje poprawność działania tych programów; – wskazuje instrukcje i rozkazy realizujące istotę metody.	– proponuje inne od przykładowych rozwiązania programowe; – układa programy realizujące algorytmy sortowania bąbelkowego i przez wstawianie według własnego pomysłu.
Fibonacci i jego wzór, czyli generujemy kolejne liczby ciągu	– wie, kim był i kiedy żył Fibonacci; – zna jego zasługi dla rozwoju cywilizacji.	– na podstawie podręcz- nika lub innych wiarygod- nych źródeł omawia na przykładzie istotę ciągu Fibonacciego; – na podstawie podręcz- nika lub innych wiarygod- nych źródeł analizuje działanie przykładowego algorytmu obliczającego kolejne elementy ciągu, zapisanego w postaci schematu blokowego.	– samodzielnie omawia na przykładzie istotę ciągu Fibonacciego; – samodzielnie analizuje działanie przykładowego algorytmu obliczającego kolejne elementy ciągu, zapisanego w postaci schematu blokowego; – układa algorytm obliczający określoną liczbę liczb ciągu Fibonacciego; – analizuje program w języku C++ ułożony według przykładowego algorytmu.	– układa program w języku C++ obliczający n kolejnych elementów ciągu Fibonacciego; – testuje poprawność działania swojego programu na przykładzie.	– proponuje rozwiązanie problemu obliczania n-tego elementu ciągu Fibonacciego.
Szukamy lidera, czyli porządkowanie zbioru metodą binarnego wyszukiwania	– omawia, na czym polega metoda dziel i zwyciężaj; – rozumie, na czym polega umieszczanie danych w	– na podstawie podręcz- nika opisuje zastosowanie wyszukiwania binarnego	– omawia na przykładzie działanie algorytmu sortowania zbioru z wykorzystaniem	– testuje algorytm sortowania zbioru z wykorzystaniem wyszukiwania binarnego	– stosuje algorytm dla tablicy wielowymiarowej.

	tablicy.	w sortowaniu.	wyszukiwania binarnego.	na dowolnych danych; – układa pogram na podstawie algorytmu.	
Przecinając oś, czyli wyznaczenie miejsc zerowych funkcji metodą połowienia	– podaje definicję miejsca zerowego funkcji i wskazuje je na wykresie.	– na podstawie podręcznika omawia metodę połowienia.	– analizuje algorytm metody połowienia z podręcznika.	– na podstawie opisu metody opracowuje algorytm metody połowienia i na jego podstawie układa program w języku C++; – testuje algorytm i program.	– omawia inne metody znajdowania miejsca zerowego, np. metodę analityczną.
Prawie robi różnicę, czyli obliczanie przybliżonej wartości pierwiastka kwadratowego	– podaje, czym jest przybliżenie wartości pierwiastka i zna konsekwencje jego stosowania.	– na podstawie podręcznika omawia metodę Newtona-Robsona.	– omawia podstawy matematyczne metody Newtona-Robsona; – analizuje omówiony w podręczniku algorytm i program metody Newtona-Robsona.	– na podstawie opisu metody samodzielnie opracowuje algorytm metody Newtona--Robsona; – układa program znajdujący przybliżoną wartość pierwiastka kwadratowego i go testuje.	– omawia inne metody obliczania wartości pierwiastka kwadratowego.
Szybkie potęgowanie liczb w wersji iteracyjnej, czyli jak obliczyć potęgę o wykładniku naturalnym	– podaje właściwości potęgi liczby 0; – własnymi słowami opisuje, czym jest potęgowanie liczb i jak się je wykonuje; – uruchamia program z podręcznika i testuje jego działanie; – podaje, na czym polega iteracyjne podejście do rozwiązywania problemów.	– na podstawie podręcznika omawia algorytm iteracyjny metody szybkiej potęgowania i porównuje ją z metodą kolejnego mnożenia przez podstawę potęgi.	– samodzielnie omawia metodę szybkiego potęgowania w wersji iteracyjnej; – analizuje algorytm i program w języku C++ metody szybkiego potęgowania.	– omawia na przykładzie zalety metody szybkiego potęgowania; – układa algorytm metody szybkiego potęgowania i na jego podstawie program w języku C++ w wersji iteracyjnej.	– porównuje podejście iteracyjne i rekurencyjne do obliczania potęgi metodą szybką.
Szybkie potęgowanie liczb w wersji rekurencyjnej, czyli jak obliczyć potęgę o wykładniku naturalnym	– uruchamia program z podręcznika i testuje jego działanie; – podaje, na czym polega	– na podstawie podręcznika omawia rekurencyjne podejście do metody potęgowania	– samodzielnie omawia metodę szybkiego potęgowania w wersji rekurencyjnej;	– układa program metody szybkiego potęgowania i na jego podstawie program w języku C++ lub	– porównuje podejście iteracyjne i rekurencyjne do obliczania potęgi metodą szybką.

	rekurencyjne podejście do rozwiązywania problemów.	szybkiego.	– analizuje algorytm i program w języku C++ lub Java dla metody szybkiego potęgowania rekurencyjnego.	Java w wersji rekurencyjnej.	
Suma binarna, czyli dodawanie w systemie dwójkowym	– omawia podstawy systemu dwójkowego; – dodaje do siebie pisemnie niewielkie liczby binarne.	– zamienia reprezentacje liczb binarnych na dziesiętne i odwrotnie.	– układa algorytm dodawania binarnego i go testuje.	– układa program w języku C++ realizujący algorytm dodawania liczb binarnych i go testuje.	– układa program na podstawie algorytmu także w innym języku (np. Java).
Różnica binarna, czyli odejmowanie w systemie dwójkowym	– odejmuje pisemnie niewielkie liczby binarne.	– opisuje metodę odejmowania liczb binarnych i tłumaczy rolę pożyczki na przykładzie.	– układa algorytm odejmowania binarnego i go testuje.	– układa program w języku C++ realizujący algorytm odejmowania liczb binarnych i go testuje.	– układa program na podstawie algorytmu także w innym języku (np. Java).
Iloczyn binarny, czyli mnożenie w systemie dwójkowym	– mnoży pisemnie niewielkie liczby binarne.	– opisuje własnymi słowami na przykładzie metodę mnożenia liczb binarnych.	– układa algorytm mnożenia binarnego i go testuje.	– układa program w języku C++ realizujący algorytm mnożenia liczb binarnych i go testuje.	– układa program na podstawie algorytmu także w innym języku (np. Java).
Iloraz binarny, czyli dzielenie w systemie dwójkowym	– dzieli pisemnie niewielkie liczby binarne.	– opisuje własnymi słowami na przykładzie metodę dzielenia liczb binarnych.	– układa algorytm dzielenia binarnego i go testuje.	– układa program w języku C++ realizujący algorytm dzielenia liczb binarnych i go testuje.	– układa program na podstawie algorytmu także w innym języku (np. Java).
Jak to zapisać, czyli zamiana systemów liczbowych z ósemkowego na szesnastkowy	– charakteryzuje liczby szesnastkowe i ósemkowe – wie, jak są zbudowane pozycyjne systemy liczbowe	– na przykładzie z podręcznika omawia działanie algorytmów zamiany postaci liczby dziesiętnej na ósemkową, dziesiętnej na szesnastkową i ósemkowej na szesnastkową – na przykładzie z podręcznika omawia działanie funkcji zamieniających postaci liczb	– na podstawie opisu i algorytmów z podręcznika układa funkcje zamieniające prezentację liczb z dziesiętnej na ósemkową i szesnastkową oraz z ósemkowej na szesnastkową	– samodzielnie układa programy (funkcje) zamieniające prezentacje liczb – układa programy wykorzystujące funkcje zamieniające prezentacje liczb	– samodzielnie układa uniwersalne programy oparte o funkcje zamieniające prezentacje liczb na wiele postaci

Mniej lub więcej ósemek, czyli dodawanie i odejmowanie w systemie ósemkowym	– wskazuje podobieństwo pomiędzy pisemnym dodawaniem i odejmowaniem liczb dziesiętnych i liczb ósemkowych	– na podstawie podręcznika omawia algorytm dodawania i odejmowania liczb ósemkowych	– na podstawie algorytmu i przykładu z podręcznika układa program dodający lub odejmujący liczby w postaci ósemkowej	– samodzielnie układa funkcjonalny program dodający i odejmujący liczby podane w systemie ósemkowym	– samodzielnie opracowuje program typu kalkulator wykonujący dodawanie lub odejmowanie liczb w systemach ósemkowym, szesnastkowym i dziesiętnym, wyświetlając wyniki w każdym z nich
Iloczyn i iloraz oktalny, czyli mnożenie i dzielenie w systemie ósemkowym	– wskazuje podobieństwo pomiędzy pisemnym mnożeniem i dzieleniem liczb dziesiętnych i liczb ósemkowych	– na podstawie podręcznika omawia algorytm mnożenia i dzielenia liczb ósemkowych	– na podstawie algorytmu i przykładu z podręcznika układa program mnożący lub dzielący liczby w postaci ósemkowej	– samodzielnie układa funkcjonalny program mnożący i dzielący liczby podane w systemie ósemkowym	– samodzielnie opracowuje program typu kalkulator wykonujący cztery podstawowe działania w systemach ósemkowym, szesnastkowym i dziesiętnym, wyświetlając wyniki w każdym z nich
Nie tylko dwójkowy i dziesiętny, czyli suma i różnica w systemie innym niż dziesiętnym	– wie, jak są zbudowane pozycyjne systemy liczbowe – umie przedstawić liczbę w dowolnym systemie pozycyjnym, np. siódemkowym	– formułuje ogólną zasadę budowy pozycyjnych systemów liczbowych	– omawia ogólne wzory podane w podręczniku dotyczące dodawania i odejmowania liczb w systemach pozycyjnych	– układa algorytm na podstawie wzorów z podręcznika dodający lub odejmujący liczby w dowolnym liczbowym systemie pozycyjnym	– układa program dodający lub odejmujący liczby w dowolnym liczbowym systemie pozycyjnym
Iloczyn i iloraz w systemie innym niż dziesiętnym, czyli wielokrotności w systemach pozycyjnych	– wie, jak są zbudowane pozycyjne systemy liczbowe – na podstawie wiedzy z poprzedniej lekcji i podręcznika formułuje zasady mnożenia i dzielenia liczb w różnych systemach liczbowych	– umie utworzyć tabliczkę mnożenia w dowolnym systemie pozycyjnym, opierając się na przykładzie z podręcznika	– formułuje algorytm w postaci listy kroków, dzielący i mnożący liczby w dowolnym pozycyjnym systemie liczbowym	– układa program dzielący i mnożący liczby w dowolnym pozycyjnym systemie liczbowym	– modyfikuje wcześniej ułożony program typu kalkulator, dodając do niego inne liczbowe systemy pozycyjne

Wyciąganie elementów z listy, czyli znajdowanie najdłuższego spójnego podciągu niemalejącego	– na podstawie podręcznika objaśnia pojęcie najdłuższego podciągu rosnącego	– samodzielnie odnajduje najdłuższy podciąg rosnący w przykładowym ciągu	– samodzielnie bada, czy dany podciąg jest najdłuższym podciągiem rosnącym danego ciągu – tworzy algorytm sprawdzania, czy dany podciąg jest rosnący – omawia algorytm wyszukujący najdłuższy podciąg rosnący	– samodzielnie tworzy algorytm wyszukujący najdłuższy podciąg rosnący i omawia istotę jego działania	– układa program wyszukujący w zbiorze najdłuższy podciąg rosnący
Elementy występujące kolejno po sobie, czyli znajdowanie najdłuższego spójnego podciągu o największej sumie	– na podstawie podręcznika objaśnia pojęcie najdłuższego spójnego podciągu o największej sumie	– samodzielnie odnajduje najdłuższy spójny podciąg o największej sumie w podanym ciągu – omawia istotę metody naiwnej – na podstawie opisu z podręcznika objaśnia istotę sum prefiksowych	– na podstawie podręcznika omawia metodę naiwną lub z zastosowaniem sum prefiksowych wyszukiwania najdłuższego spójnego podciągu o największej sumie	– samodzielnie tworzy algorytm i program wyszukujący najdłuższy spójny podciąg o największej sumie na podstawie metody naiwnej lub stosując sumy prefiksowe	– samodzielnie układa programy dla obu metod
Ta sama kolejność, czyli szukanie najdłuższego wspólnego podciągu	– omawia istotę metod opartych na rekurencji – umie znaleźć w przykładzie najdłuższy wspólny podciąg	– samodzielnie odnajduje najdłuższy wspólny podciąg dla przykładowych ciągów – na podstawie podręcznika omawia algorytm dynamiczny i oparty na rekurencji rozwiązujący problem odnajdowania najdłuższego wspólnego podciągu	– samodzielnie bada, czy dany podciąg jest najdłuższym wspólnym podciągiem dwóch ciągów – na podstawie podręcznika omawia zastosowanie metody rekurencyjnej i dynamicznej	– układa program wyszukujący NWP metodą rekurencyjną	– układa programy, stosując obie metody – rekurencyjną i dynamiczną