Roczny plan dydaktyczny przedmiotu informatyka dla klasy IV liceum ogólnokształcącego i technikum w zakresie rozszerzonym, uwzględniający kształcone umiejętności i treści podstawy programowej.

UWAGA! Założono, że:
– w pracowni wykorzystuje się komputery uczniowskie, podręcznik, komputer nauczyciela z projektorem lub ekranem lub tablicę interaktywną, platformę e-learningową do udostępniania plików i przesyłania przez uczniów prac domowych (nie jest niezbędne do realizacji planu), dlatego nie wymienia się ich w kolumnie „Propozycje środków dydaktycznych";
– nauczyciel korzysta z aplikacji ze scenariuszami lekcji (do realizacji planu nie jest ona niezbędna);
– nauczyciel może dowolnie modyfikować wszystkie elementy niniejszego planu dydaktycznego.

Temat (rozumiany jako lekcja)	**Liczba godzin**	**Treści podstawy programowej**	**Cele ogólne**	**Kształcone umiejętności**	**Propozycje metod nauczania**	**Propozycje środków dydaktycznych**	**Uwagi**
I Algorytmika i programowanie							
1. Więcej o liczbach, czyli logarytm, sito Eratostenesa i schemat Hornera.	2 – 3	II.R 1c. II.R 1c.	– przypomnienie definicji liczby pierwszej – nabycie umiejętności generowania liczb pierwszych dla zadanego przedziału – poznanie metody obliczania wartości wielomianu w punkcie	– przypomnienie pojęcia logarytmu – poznanie algorytmu sita Eratostenesa do generowania listy wszystkich liczb pierwszych w zadanym przedziale – nauczenie się schematu Hornera – prostej metody obliczania wartości wielomianu	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Pojęcie logarytmu – definicja matematyczna, analogie z liczbą cyfr w zapisie dwójkowym. Inne analogie (turniej systemem pucharowym). Uczniowie podają definicje liczby pierwszej i sposoby sprawdzania, czy liczba jest pierwsza. Dyskusja o złożoności (jak zachowa się tradycyjny algorytm „pierwiastkowy" na

							wielocyfrowych liczbach, gdzie leży granica jego stosowania). Przejście do problemu znalezienia wszystkich liczb z zadanego przedziału) – nauczyciel pokazuję algorytm sita Eratostenesa. Schemat Hornera – zwrócenie uwagi, że używaliśmy schematu Hornera do przeliczania wartości z różnych pozycyjnych systemów liczbowych. Samodzielna implementacja przez uczniów wybranych przez nauczyciela algorytmów z lekcji.
2. Rekursja raz jeszcze, czyli dziel i zwyciężaj.	2 – 3	IR.8 IIR. 1a. II.R 1i. IIR. 3a. IIR. 3c.	– zrozumienie pojęcia rekursji – warunki działania algorytmów rekurencyjnych – umiejętność stosowania rekursji do rozwiązywania problemów – omówienie metody dziel-i-zwyciężaj.	– użycie rekursji do policzenia sumy cyfr – użycie rekursji do potęgowania – użycie prostego w implementacji rekurencyjnego wariantu algorytmu szybkiego potęgowania	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Nauczyciel wyjaśnia krótko, czym jest rekurencja. Uczniowie zastanawiają się, jak napisać rekurencyjnie znany im problem policzenia sumy cyfr i podnoszenia do potęgi. Inne proste przykłady funkcji i programów rekurencyjnych, również proponowane przez uczniów. Warunki działania rekursji, analiza błędnych kodów (bez warunku początkowego,

							niezmniejszające danych etc.) Rekursja z dwoma (lub więcej) wywołaniami – metoda dziel-i-zwyciężaj. Algorytm potęgowania w wersji iteracyjnej, liniowej rekurencyjnej. Analiza złożoności, przejście do złożoności logarytmicznej poprzez dzielenie przez 2. Samodzielna implementacja przez uczniów algorytmu szybkiego potęgowania.
3. Sortowanie przez scalanie, czyli pierwszy efektywny algorytm sortowania.	2	II.R 1e. II.R 3b.	– poznanie i zrozumienie algorytmu sortowania przez scalanie	– poznanie sposobu łączenia dwóch posortowanych tablic w jedną – wykorzystanie tej procedury do rekurencyjnego sortowania dowolnej tablicy – zaimplementowanie tego algorytmu – określenie złożoności obliczeniowej algorytmu sortowania przez scalanie	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Uczniowie przypominają sobie znane im algorytmy sortowania w czasie O(n^2) – sortowanie przez wstawianie/selekcję/bąbelkowe. Uczniowie zastanawiają się, jak scalić 2 posortowane tablice. Nauczyciel wyjaśnia, na czym polega sortowanie przez scalanie. Potencjalnie: analiza złożoności (najpierw na małych przykładach tablic 8–16-elementowych, potem ogólna); samodzielna implementacja algorytmu sortowania przez scalanie (zalecane użycie typu

							vector w C++ lub listy w Pythonie).
4. Sortowanie szybkie, czyli jak się sortuje w praktyce.	2	II.R 3b.	– poznanie i zrozumienie algorytmu sortowania szybkiego	– poznanie algorytmu sortowania szybkiego – zaimplementowanie tego algorytmu – określenie złożoności obliczeniowej algorytmu sortowania szybkiego	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Nauczyciel wyjaśnia, na czym polega sortowanie szybkie – najpierw tylko operacja podziału tablicy z danym kluczem, potem cała procedura z wywołaniami rekurencyjnymi. Potencjalnie: dyskusja, jak efektywnie wybrać klucz i jaki to ma wpływ na złożoność; samodzielna implementacja algorytmu QuickSort. Funkcje sort() z biblioteki standardowej w C++/Pythonie.
5. Specjalne elementy w tablicy, czyli największy, najmniejszy i lider.	2	II.R 1b. II.R 1c. IIR. 3c	– przypomnienie, jak szukać najmniejszego i największego elementu w tablicy – zrozumienie pojęcia lidera i dominanty – poznanie sposobu na efektywne szukanie elementów specjalnych w tablicy	– poznanie i zrozumienie sposobu na to, jak zaoszczędzić na liczbie operacji przy szukaniu jednocześnie elementu największego i najmniejszego w tablicy – poznanie i zrozumienie algorytmu szukania dominanty oraz algorytmu szukania lidera	Dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Uczniowie przypominają, jak znajduje się w tablicy element najmniejszy bądź największy. Ile porównań wymaga znalezienie obu tych elementów? (w prostej wersji: 2n). Algorytm jednoczesnego znajdowania z 3/2 n porównaniami. Samodzielna implementacja algorytmu. Algorytmy wyszukiwania dominanty i lidera – propozycje od uczniów (jest wiele rozwiązań tylko nieznacznie gorszych od

							optymalnego, dostępnych do wymyślenia dla uczniów). Nauczyciel wyjaśnia algorytm optymalny, znajdujący lidera „w jednym przelocie”.
6. Tablice dynamiczne i listy wiązane, czyli kiedy tablica nie wystarcza.	2	IR.8 IIR.1 IIR. 3i	– poznanie, czym są struktury danych – nabycie umiejętności wybierania odpowiedniej struktury	– zrozumienie, czym jest tablica – niezależnie od języka programowania – zrozumienie, czym jest struktura danych – poznanie różnych sposobów przechowywania danych, analogiczne do tablicy, lecz pozwalające na inne sposoby dostępu do nich: tablicy dynamicznej (wektora) oraz listy wiązanej – poznanie problemu Flawiusza i sposobu jego rozwiązania z użyciem dynamicznych struktur danych	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Nauczyciel wyjaśnia, czym jest tablica. Dyskusja o wadach tablicy – np. niemożliwość usunięcia elementu „ze środka” (a także np. szybkiego wyszukiwania). Potencjalne inny sposoby zapisu danych. Nauczyciel wyjaśnia pojęcie tablicy dynamicznej, uczniowie mogą przyporządkować je do już znanych typów danych w C++/Pythonie (wektory/listy). Lista wiązana, operacje na niej. Problem Flawiusza i rozwiązanie go za pomocą listy. Potencjalnie: próba zdefiniowania pojęcia „struktury danych”, próba użycia tablic dynamicznych i list w C++/Pythonie.
7. Stos i kolejka, czyli struktury proste i bardzo użyteczne.	2 – 3	IR.8 II.R. 1	– poznanie, czym są struktury stosu i kolejki	– przypomnienie, czym są struktury danych – poznanie różnych sposobów przechowywania danych,	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Struktura stosu – zestaw możliwych operacji, kolejność elementów na stosie. Wspólna analiza

		IIR. 3i.		analogiczne do tablicy, lecz pozwalające na inne sposoby dostępu do nich: lista i kolejka – zdefiniowanie podstawowych funkcji dla stos i kolejki – poznanie przykładów zastosowania stosu i kolejki – nabycie umiejętności implementowania stosu i kolejki – poznanie bibliotek, w których są już gotowe implementacje stosu i kolejki – nabycie umiejętności używania stosu i kolejki w programach, np. do zaimplementowania odwrotnej notacji polskiej			przykładowego ciągu operacji na stosie. Implementacja stosu – typy dostępne w bibliotece standardowej, ewentualnie samodzielna implementacja za pomocą tablicy/listy. Przypomnienie algorytmu ONP i użycie w nim stosu, ewentualna samodzielna implementacja algorytmu ONP. Kolejka – analogiczne wprowadzenie do operacji, omówienie kolejności wstawiania i usuwania elementów, analiza przykładowych ciągów operacji.
8. Grafy, czyli co mają wspólnego sieć społecznościowa i paryskie metro.	1	IIR. 3j	– poznanie, czym jest graf i jak reprezentować graf w pamięci komputera	– poznanie różnych przykładów grafów w życiu codziennym (skierowanych i nieskierowanych) – nabycie umiejętności reprezentowania grafu za pomocą macierzy sąsiedztwa i listy sąsiedztwa	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Omówienie podanych przykładów sytuacji, w których występują grafy. (Uczniowie mogą też zgłaszać własne propozycje). Formalna definicja pojęcia grafu. Wyjaśnienie implementacji przez macierze sąsiedztwa i przez listy sąsiedztwa. Napisanie macierzy/list sąsiedztwa dla przykładowych grafów narysowanych na tablicy.

9. Wyszukiwanie idola, czyli o tym, jak spóźniliśmy się na turniej tenisowy.	1	II.R 1b	– przedstawienie turnieju za pomocą grafu skierowanego – poznanie definicji idola	– poznanie, czym jest turniej i jaki specjalny wierzchołek nazywa się idolem – poznanie algorytmu wyszukiwania idola, który nie musi znać całego grafu	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Narysowanie przykładowych grafów turniejów – takich, w których jest idol, i takich, w których nie ma. Możliwa dyskusja z uczniami – jak znaleźć idola, nie znając całego grafu, zadając możliwie najmniej pytań? Jeśli uczniowie nie dojdą sami do algorytmu, objaśnia go nauczyciel. Analiza złożoności (czyli: ile pytań jest w najgorszym wypadku potrzebnych do znalezienia idola).
10. Najkrótsze ścieżki w grafie, czyli jak komputery znajdują drogę.	1 – 2	IIR. 3j	– poznanie i zrozumienie sposobów znajdowania najkrótszej ścieżki w grafie	– algorytm BFS (przeszukiwania grafu wszerz), służącego do znajdowania najkrótszych ścieżek w grafie	Wykład, dyskusja i ćwiczenia.	Wybrane przez nauczyciela środowisko programistyczne.	Najkrótsza ścieżka w grafie, analogie do sytuacji praktycznych. Algorytm BFS dokładnie omówiony na kilku małych przykładach grafów. Ewentualnie: samodzielna implementacja algorytmu.
11. Programowanie strukturalne kontra obiektowe, czyli dane w centrum uwagi.	1	II.R. 1 II.R. 2	– poznanie i zrozumienie programowania obiektowego	– poznanie nowego sposobu myślenia o pisaniu programów: programowanie obiektowe, w którym centralnym pojęciem nie są procedury, lecz typy danych (klasy) i zmienne (obiekty) – poznanie podstawowych pojęć programowania obiektowego: klasa, obiekt, pole, metoda, a także – w dużym skrócie – mechanizmy dziedziczenia i polimorfizmu	Wykład, dyskusja.	Wybrane przez nauczyciela środowisko programistyczne.	Nauczyciel wyjaśnia ideę programowania obiektowego, omawia różnice między paradygmatem strukturalnym a obiektowym. Wspólna dyskusja o zaletach i wadach programowania obiektowego. Ewentualnie: próba

							bardzo ogólnego „zaplanowania”, jak wyglądałby większy projekt programistyczny (gra wideo, edytor tekstu, komunikator) i jak go dzielić między programistów.
12. Jak rozwiązywać zadania programistyczne, czyli podstawowe porady na proste kody.	1	I.1 IV.3	– poznanie i przemyślenie dobrych praktyk przy programowaniu i szukaniu błędów w kodach programów	– nabycie wiedzy, jak wyglądają typowe zadania algorytmiczno-programistyczne (zarówno maturalne, jak i na wszelkiego rodzaju konkursach) – poznanie kilku podstawowych porad, jak podchodzić do takich zadań, i ogólnie – do programowania	Dyskusja.	Wybrane przez nauczyciela środowisko programistyczne.	Nauczyciel krótko omawia porady podane w podręczniku. Wspólna dyskusja o sensowności pisania czytelnych kodów – porównanie praktyk programistycznych, które stosują uczniowie.
II. Arkusz kalkulacyjny							
13. Podstawowe typy danych, czyli co możemy wpisać w komórkę arkusza.	1	II.P. 3c II.R 4c.	– przypomnienie, jakie dane można wpisać w komórkę arkusza; – przypomnienie sposobów adresowania komórek – przypomnienie podstawowych funkcji używanych w formułach	– nabycie umiejętności rozróżniania typów danych wpisanych w komórkę – rozróżnianie adresowania względnego, bezwzględnego i mieszanego – przypomnienie podstawowych funkcji	Krótka dyskusja przypominająca wiadomości z arkusza w poprzednich lat. Ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel zwraca uwagę na problemy ze złym formatowaniem komórki (np. wpisanie liczby w komórkę z ustawianym formatowaniem jako tekst).
14. Trójkąt Sierpińskiego, czyli do czego przydaje się adresowanie względne i adresowanie bezwzględne oraz	2	II.P. 3c II.R 4c. II.R 1k IV.5	– praktyczne wykorzystanie właściwości arkusza kalkulacyjnego do prostych symulacji fraktali	– wykorzystanie w praktyce różnego adresowania komórek – poznanie sposobu szybkiego wypełniania komórki podobnymi wartościami, – przypomnienie, jak korzystać z funkcji jeżeli	Wykład i ćwiczenia.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel pokazuje i dokładnie omawia, jak wygenerować trójkąt Sierpińskiego w arkuszu. Uczniowie samodzielnie generują fraktal z ćwiczeń.

wypełnienie serią danych.				– poznanie funkcji los.zakr() – nabycie umiejętności szybkiego kopiowania i zaznaczania danych – przypomnienie, jak wstawić wykres punktowy			
15. Jak korzystać z gotowych funkcji, podstawowe funkcje tekstowe, czyli jak łatwo modyfikować dane tekstowe.	1	II.P. 3c II.R 4c.	– poznanie wybranych funkcji tekstowych arkusza kalkulacyjnego	– nabycie umiejętności wykorzystania funkcji dł(), lewy(), prawy(), fragment.tekstu(), znajdź(), złacz.tekst(), wielkie.litery(), tekst() – nabycie umiejętności samodzielnego wybrania i używania funkcji tekstowych w listy wszystkich funkcji	Ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel pokazuje przykłady użycia wybranych funkcji. Uczniowie samodzielnie wykonują ćwiczenia.
16. Jak korzystać z gotowych funkcji, podstawowe funkcje dotyczące daty i czasu, czyli jak łatwo pracować z datą.	1	II.P. 3c II.R 4c.	– poznanie wybranych funkcji daty i czasu arkusza kalkulacyjnego	– nabycie umiejętności wykorzystania funkcji rok(), miesiąc(), dzień(), teraz(), dni.robocze(), godzina(), minute(), sekunda() – nabycie umiejętności samodzielnego wybrania i używania funkcji operujących na dacie i czasie w listy wszystkich funkcji	Ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel mocno podkreśla, czym jest data w arkuszu kalkulacyjnym (data to liczba) i jakie są tego konsekwencje. Uczniowie wykonują ćwiczenia.
17. Jak szukać w Excelu, czyli gdzie to jest.	1	II.P. 3c II.R 4c.	– nabycie umiejętności wyszukiwania w arkuszu kalkulacyjnym	– nabycie umiejętności wyszukiwać wartości, korzystając z funkcji WYSZUKAJ – poznanie funkcji WYSZUKAJ.PIONOWO, WYSZUKAJ.POZIOMO – nabycie umiejętności samodzielnego czytania dokumentacji do jej użycia	Ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel pokazuje, jak używać funkcji WYSZUKAJ. Uczniowie samodzielnie zdobywają informacje, jak korzystać z funkcji WYSZUKAJ.PIONOWO.
18. Co ukrywa numer PESEL, czyli do czego przydaje się adres mieszany.	2	II.P. 3c II.R 4c.	– poznanie co to jest numer PESEL i jakie informacje można z niego uzyskać	– nabycie umiejętności wyznaczania daty urodzin i płci z numeru PESEL – nabycie umiejętności sprawdzania, czy numer PESEL jest poprawny	Pokaz i ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Uczniowie wykonują zadane ćwiczenia. Nauczyciel przypomina o szybkim zaznaczaniu i kopiowaniu.

			– nabycie biegłości w wykorzystaniu adresu mieszanego	– poznanie, jak szybko konwertować liczbę na tekst i odwrotnie w arkuszu kalkulacyjnym			
19. Tabela przestawna, czyli wygodne grupowanie danych.	1	II.P. 3c II.R 4c.	– nabycie umiejętności grupowania danych według zadanego klucza i używania funkcji liczących w grupach	– poznanie i używanie tabeli przestawnej jako narzędzia do grupowania danych w arkuszu kalkulacyjnym	Pokaz i ćwiczenia.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel pokazuje przykład użycia. Uczniowie wykonują ćwiczenia.
20. Wykresy, czyli jak atrakcyjnie przedstawiać dane.	1	II.P. 3c II.R 4c.	– nabycie umiejętności przedstawiania danych za pomocą wykresów	– poznanie podstawowych typów wykresów – nabycie umiejętności wyboru typu wykresu do potrzeb – nabycie umiejętności tworzenia wykresu w arkuszu – nabycie umiejętności pracy z wykresem (opis osi, zmiana skali…)	Pokaz i ćwiczenie.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel prosi uczniów o przypomnienie, jakie typy wykresów znają i do czego można ich używać. Uczniowie wykonują ćwiczenie.
21. Pobieranie danych z pliku, czyli z notatnika do Excela.	1	II.P. 3c II.R 4c.	– nabycie umiejętności pobierania danych do arkusza z pliku tekstowego	– poznanie różnych sposobów pobierania danych z pliku tekstowego do arkusza	Pokaz i ćwiczenia.	Arkusz kalkulacyjny z pakietu biurowego lub chmury, np. Office 365.	Nauczyciel zwraca uwagę, o czym należy pamiętać, importując dane z pliku do arkusza (separator kolumn, numer PESEL, liczby zmiennoprzecinkowe). Uczniowie wykonują ćwiczenia.