2020/2021
M. Tomaben

Zasady Przedmiotowego Oceniania (PZO) – programowanie aplikacji. Klasa IV TI.

I Zasady systemu oceniania

1. Ocenianie pracy uczniów odbywa się na podstawie przeprowadzonych
   - sprawdzianów praktycznych,
   - sprawdzianów teoretycznych
   - kartkówek,
   - odpowiedzi ustnych,
   - prac domowych
   - zadań dodatkowych
   - aktywności uczniów na lekcji.
2. Sprawdziany są obowiązkowe.
3. Oceny ze sprawdzianów stanowią najważniejsza część składowa oceny semestralnej (rocznej).
4. Sprawdziany są zapowiadane z co najmniej tygodniowym wyprzedzeniem.
5. Uczeń ma prawo przystąpić do sprawdzianu powtórnie (pisemnie lub ustnie) tylko raz w ciągu dwóch tygodni od daty zapoznania się z oceną (w terminie uzgodnionym z nauczycielem).
6. Uczeń oceniany jest za swoje osiągnięcia w nauce (wiedza i umiejętności) oraz postawę (aktywność i kreatywność) - w przypadku stwierdzenie niesamodzielności pracy uczeń otrzymuje ocenę niedostateczną i traci prawo do poprawy tej oceny.
7. Uczeń nieobecny na sprawdzianie z przyczyn losowych ma obowiązek przystąpić do sprawdzianu w terminie wskazanym przez nauczyciela.
8. Uczeń może otrzymać ocenę dodatkową za udział w konkursach i olimpiadach przedmiotowych oraz projektach.
9. Prowadzenie zeszytu (teczki z wydrukami) jest obowiązkowe.
10. W przypadku oceny niedostatecznej na I semestr uczeń ma obowiązek wykazania się opanowaniem wiadomości i umiejętności z zakresu treści programowej I semestru w ciągu 30 dni od dnia zakończenia I semestru, w terminie uzgodnionym z nauczycielem. Uzyskane oceny są wpisywane jako oceny cząstkowe II semestru. Zlekceważenie tego obowiązku upoważnia nauczyciela do wpisania do dziennika oceny niedostatecznej,
10. W innych sprawach zastosowanie ma ZWO.

Stopień niedostateczny
Uczeń nie opanował podstawowych wiadomości i umiejętności niezbędnych do dalszego zdobywania wiedzy, nie rozwiązuje najprostszych zadań z pomocą nauczyciela, nie wykazuje zainteresowania treściami prezentowanymi na lekcjach, nie rozwiązuje ćwiczeń, zadań domowych, otrzymuje cząstkowe oceny niedostateczne, których nie poprawia.

Stopień dopuszczający (poziom wymagań K - konieczny)
Uczeń naprowadzony przez nauczyciela potrafi rozwiązywać proste (jednoetapowe) problemy (z zakresu określonego w planie wynikowym), które już wcześniej rozwiązywał, wykorzystując do tego celu poznane oprogramowanie. Potrafi zapisać wyniki swojej pracy na wskazanym nośniku.
Ocena dopuszczająca, to ocena dla ucznia słabego, który źle radzi sobie z pracą przy komputerze, nie potrafi samodzielnie wykonać ćwiczenia i nie w pełni rozumie zadanie przed nim postawione, który przy pomocy nauczyciela umie jednak zrealizować minimum ustalone dla danego ćwiczenia. W jego poczynaniach widać duże braki w zakresie wiedzy i umiejętności, ale podejmuje on próby zmierzenia się z zadaniem.

Stopień dostateczny (poziom wymagań P - podstawowy)
Uczeń potrafi dobrać podstawowe narzędzia programowe do rozwiązania prostego problemu z zakresu określonego w planie wynikowym. Samodzielnie rozwiązuje problemy realizowane w trakcie nauki, bazując na opanowanych podstawowych wiadomościach. W wykonywanie zadania wkłada dużo pracy i wysiłku.
Ocenę dostateczną otrzymuje uczeń wykazujący braki w umiejętnościach i wiedzy, nadrabiający jednak pracowitością i chęcią wykonania ćwiczenia. Uczeń wykonuje swoją pracę poprawnie pod względem użycia funkcji programu, ale mało estetycznie i z błędami. Projekt pozostanie niewykończony. Uczeń stosuje jedynie podstawowe funkcje oprogramowania.

Stopień dobry (poziom wymagań R - rozszerzający)
Uczeń potrafi samodzielnie dobrać podstawowe narzędzia programowe do rozwiązania kilkuetapowego problemu (wymagającego wykorzystania kilku narzędzi). Rozwiązuje problemy (określone w planie wynikowym) o tym samym stopniu trudności, co realizowane w ramach programu nauczania.
Ocena dobra jest oceną dla ucznia samodzielnie wykonującego ćwiczenia, którego prace zawierają drobne błędy, lecz są wykonane estetycznie. Uczeń wykazuje znajomość programu i jego średnio zaawansowanych funkcji. Stosuje klasyczne rozwiązania, wzorowane na istniejących projektach.

Stopień bardzo dobry (poziom wymagań D - dopełniający)
Uczeń wykorzystując znane narzędzia programowe, potrafi rozwiązać złożone problemy z zakresu określonego w planie wynikowym, z którymi do tej pory nie zetknął się na lekcji. Przygotowuje i prowadzi prelekcje oraz referaty związane z tematyką zajęć.
Ocenę bardzo dobrą stawiamy uczniowi biegle posługującemu się oprogramowaniem i urządzeniami peryferyjnymi, dobrze dobierającemu materiał do projektów, umiejącemu zaproponować kilka alternatywnych rozwiązań, wykonującemu projekt bezbłędnie i estetycznie.

Stopień celujący (poziom wymagań W - wykraczający)
Uczeń wykorzystując dostępne narzędzia programowe, potrafi samodzielnie rozwiązać złożone (nietypowe) problemy z zakresu określonego w planie wynikowym. Formułuje problemy w kategoriach informatyki. Bierze czynny udział w konkursach i olimpiadach informatycznych, zdobywając punktowane miejsca. Przygotowuje i prowadzi prelekcje oraz referaty poszerzające tematykę zajęć, wykazując się wiadomościami wykraczającymi poza program nauczania. Potrafi rozwijać swoje uzdolnienia.
Na **ocenę celującą** zasługuje uczeń, który w czasie ćwiczenia stosuje zaawansowane funkcje programu i sprzętu nieomawiane na zajęciach (wykraczające ponad wymagania programowe), który wykona projekt na dobrym poziomie, estetyczny, dobrze skomponowany. Uczeń umie także zaproponować własne, oryginalne pomysły, a jego projekty są funkcjonalne i wykończone.

Ocena	Wymagania dla ucznia
Niedostateczny	Uczeń nie opanował podstawowych wiadomości i umiejętności niezbędnych do dalszego zdobywania wiedzy, nie rozwiązuje najprostszych zadań z pomocą nauczyciela, nie wykazuje zainteresowania treściami prezentowanymi na lekcjach, nie rozwiązuje ćwiczeń, zadań domowych, otrzymuje cząstkowe oceny niedostateczne, których nie poprawia.
Dopuszczający	Zna pojęcie algorytmu, umie podać przykłady zadań algorytmicznych Potrafi zapisać prosty algorytm w pseudojęzyku lub za pomocą listy kroków Zapisuje prosty algorytm za pomocą schematu blokowego Opanował tylko najważniejsze elementy języków C++ i C#, potrafi z pomocą nauczyciela wykonać zadanie algorytmiczne o elementarnym stopniu trudności Zna pojęcie funkcji, potrafi podać poprawną ich deklarację, definicję i wywołanie. Potrafi napisać prostą funkcję. Potrafi wykonać ćwiczenia korzystając z pomocy nauczyciela Potrafi podać definicję typu tablicowego i strukturalnego. Potrafi z pomocą nauczyciela dokonać podstawowego przetwarzania danych zawartych w tablicy. Potrafi dokonać elementarnych operacji na plikach. Zna pojęcie wskaźnika, klasy. Potrafi zadeklarować i zdefiniować prostą klasę, rozpoznać, wymienić i scharakteryzować składowe klasy. Zna istotę dziedziczenia, znaczenie kwalifikatorów dostępu: protected, public i private w dziedziczeniu. Zna zasady tworzenia wzorców klas i funkcji. Potrafi skonstruować prosty program obliczeniowy w środowisku graficznym C#.
Dostateczny	Opanował podstawowe elementy języków C++ i C#, potrafi z pomocą nauczyciela utworzyć schemat blokowy lub opis algorytmu w postaci listy kroków. Potrafi napisać prostą funkcję. Zna podstawowe algorytmy numeryczne. Z pomocą nauczyciela potrafi zastosować algorytmy w sytuacjach typowych. Zna zasady przetwarzania danych w tabelach, potrafi zadeklarować i używać tablic jednowymiarowych, zna i potrafi zastosować struktury. Stosuje pliki jako źródło danych i miejsce przechowywania danych. Rozumie co to są zmienne dynamiczne i cel ich stosowania. Umie zadeklarować obiekt i odwołać się do jego składowych, rozumie działanie konstruktora i destruktora. Zna i rozumie pojęcie przeciążania operatorów. Konstruuje proste klasy pochodne, potrafi skonstruować proste funkcje wirtualne; zna istotę i zasady polimorfizmu. Potrafi skonstruować program obliczeniowy w środowisku graficznym C#.
Dobry	Opanował większość elementów języków C++ i C#. Potrafi samodzielnie rozwiązać proste zagadnienia. Potrafi zaprojektować algorytm w postaci listy kroków lub schematu blokowego Potrafi wykorzystać różne metody przekazywania parametrów. Potrafi wskazać różnice pomiędzy iteracją i rekurencją. Zna podstawowe algorytmy rekurencyjne. Potrafi przetwarzać tablice wielowymiarowe. Potrafi zorganizować bazę danych złożoną ze struktur, umie zastosować stworzone przez siebie struktury do pamiętania danych w sytuacjach typowych. Potrafi zadeklarować i zdefiniować konstruktor i destruktor, wykorzystuje obiekty klas jako składowe innych klas, jako argumenty funkcji, korzysta ze wskaźników do obiektów; tworzy tablice obiektów. Potrafi zastosować w programach funkcje i klasy zaprzyjaźnione. Potrafi konstruować klasy

	abstrakcyjne. Potrafi zastosować w programach dziedziczenie wielokrotne, Konstruuje i stosuje funkcje „"pure virtual". Stosuje w programach kolekcje standardowe. Potrafi skonstruować program obliczeniowy w środowisku graficznym C# z obsługą wyjątków.
Bardzo dobry	Opanował wszystkie elementy języków C++ i C#. Potrafi samodzielnie rozwiązać typowe zadania programistyczne, sprawnie porusza się w środowisku programistycznym. Potrafi zastosować algorytmy rekurencyjne do rozwiązania klasycznych algorytmów. Potrafi wykorzystać tworzenie własnych modułów przy tworzeniu złożonych programów. Potrafi utworzyć program pracujący w środowisku graficznym Samodzielnie potrafi pisać programy do przetwarzania zbiorów typu baza danych, potrafi rozwiązywać zadania z użyciem struktur, umie stworzyć bazę danych. Biegle posługuje się debuggerem. Biegle tworzy algorytmy w programie JavaBlock. Stosuje przeciążanie w programach. Stosuje biegle polimorfizm, Konstruuje i stosuje w programowaniu klasy abstrakcyjne. Samodzielnie tworzy własne wzorce klas i funkcji. Biegle posługuje się STL. Biegle konstruuje programy w środowisku graficznym C#. Stosuje obsługę wyjątków i udostępnianie poleceń w programach WFA.
Celujący	Opanował wszystkie elementy języków C++ i C#, rozwiązuje zadania o dużym stopniu skomplikowania, potrafi efektywnie korzystać ze środowiska programistycznego. Rozwiązuje problemy algorytmiczne w zakresie rekurencji i iteracji w sytuacjach nowych i nietypowych. Potrafi tworzyć skomplikowane i o wysokiej estetyce programy pracujące w środowisku graficznym. Potrafi napisać aplikację bazodanową stosując poznane struktury danych, samodzielnie rozwiązuje zadania z użyciem poznanych struktur danych (np. zamiana liczby z dowolnie podanego systemu liczbowego na dowolny inny). Przejawia twórcze myślenie, proponuje niekonwencjonalne formy analiz i rozwiązań problemów programistycznych.

Planowane powtórzenia materiału oraz pisemne sprawdziany wiadomości są przeprowadzane zgodnie z zasadami określonymi w wewnątrzszkolnym systemie oceniania.
Nauczyciel w terminie ustalonym w ZWO informuje uczniów i rodziców o przewidywanych rocznych (semestralnych) ocenach. W przypadku, gdy uczeń, lub jego rodzice nie zgadzają się z przewidywaną oceną, a sprzeciw ma uzasadnienie w ocenach cząstkowych ucznia, uczeń ma prawo w terminie i formie ustalonej przez nauczyciela przystąpić do zaliczenia partii materiału objętego okresem klasyfikacji.

Oceny cząstkowe, śródroczne i końcoworoczne nauczyciel uzasadnia w formie ustnej.

Dostosowanie wymagań dla ucznia z dysleksją rozwojową.
-docenić chęć pokonywania trudności, wysiłek i wytrwałość w działaniu, samodzielność ład w miejscu pracy i porządek w działaniu
-pozostawić więcej czasu na wykonanie pracy
-stosować polecenia krótkie i nieskomplikowane
-upewnić się czy uczeń zrozumiał polecenie
-w sprawdzianach - zadaniach przeznaczonych do samodzielnego wykonania upewnić się czy uczeń zrozumiał polecenie. Zadania powinny być zapisane na kartce-nie dyktować.
-posadzić ucznia blisko nauczyciela by nauczyciel mógł kontrolować pracę ucznia

W przypadku innych dysfunkcji - szczegółowe dostosowanie wymagań – zgodnie z orzeczeniem poradni psychologiczno – pedagogicznej.

Uszczegółowione efekty kształcenia dla zawodu technik informatyk.

Efekty kształcenia z podstawy programowej Uczeń:	Uszczegółowione efekty kształcenia Uczeń po zrealizowaniu zajęć potrafi:
EE.09.1(1) stosuje podstawy algorytmiki;	EE.09.1(1)1 wyjaśnić definicję algorytmu;
	EE.09.1(1)2 zanalizować algorytmy zapisane;
	EE.09.1(1)3 scharakteryzować rodzaje algorytmów;
	EE.09.1(1)4 wykorzystać algorytmy jako rozwiązanie typowych problemów
	EE.09.1(1)4 wyszukiwać błędy w zapisanych algorytmach;

	EE.09.1(1)5 scharakteryzować algorytmy liniowe;
	EE.09.1(1)6 scharakteryzować algorytmy warunkowe;
	EE.09.1(1)7 scharakteryzować algorytmy iteracyjne z wykorzystaniem typów prostych i strukturalnych;
	EE.09.1(1)8 scharakteryzować algorytmy rekurencyjne z wykorzystaniem typów prostych i strukturalnych;
	EE.09.1(1)9 charakteryzować podstawowe algorytmy
EE.09.1(2) stosuje zasady algorytmicznego rozwiązywania problemów;	EE.09.1(2)1 charakteryzować podstawowe metody rozwiązywania problemów;
	EE.09.1(2)2 stosować podejście algorytmiczne do rozwiązywania problemów
	EE.09.1(2)3 zanalizować algorytmy w postaci schematów blokowych, listy kroków lub drzew decyzyjnych;
EE.09.1(3) stosuje podstawowe zasady programowania;	EE.09.1(3)5 zastosować programowanie zorientowane obiektowo;
	EE.09.1(3)6 zapisać algorytmy w kompilowanym języku wysokiego poziomu;
	EE.09.1(3)7 zdefiniować etapy tworzenia programu komputerowego;
	EE.09.1(3)8 zidentyfikować dane wejściowe i wyjściowe oraz pomocnicze;
	EE.09.1(3)9 zaprojektować strukturę programu pod względem niezbędnych instrukcji, procedur i funkcji (metod);
	EE.09.1(3)10 zanalizować programy (strukturę danych oraz algorytmów);
	EE.09.1(3)11 zanalizować algorytmy w postaci schematów blokowych, listy kroków lub drzew decyzyjnych;
EE.09.1(4) wykorzystuje środowisko programistyczne: edytor i kompilator;	EE.09.1(4)1 zidentyfikować różne środowiska programistyczne;
	EE.09.1(4)2 dobrać odpowiednie środowiska programistyczne do określonych zadań lub języków programowania;
	EE.09.1(4)3 przygotować do pracy różne środowiska programistyczne;
	EE.09.1(4)4 wykorzystać różne środowiska programistyczne do tworzenia aplikacji;
	EE.09.1(4)5 skompilować napisany program;
	EE.09.1(4)6 wyszukać błędy w kompilowanym programie;
EE.09.1(5) korzysta z wbudowanych typów danych;	EE.09.1(5)1 omówić podstawowe wbudowane typów danych oraz ich specyfikatorów;
	EE.09.1(5)2 zastosować wbudowane typów danych oraz ich specyfikatorów;
	EE.09.1(5)3 zadeklarować stałe i zmienne w odniesieniu do wbudowanych typów danych;

EE.09.1(6) tworzy własne typy danych;	EE.09.1(6)1 zdefiniować pojęcia dotyczące własnych typów danych (typ wyliczeniowy, unie, klasy, tablice);
	EE.09.1(6)2 zastosować deklaracje stałych i zmiennych w odniesieniu do własnych typów danych;
	EE.09.1(6)3 zidentyfikować pola i metody występujące we własnych typach danych;
	EE.09.1(6)4 stworzyć własne typy danych w wybranych językach programowania;
EE.09.1(7) stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania;	EE.09.1(7)1 zidentyfikować operatory arytmetyczne, bitowe, logiczne oraz relacji;
	EE.09.1(7)2 zidentyfikować wbudowane instrukcje, funkcje (metody), procedury i obiekty wybranych języków programowania;
	EE.09.1(7)3 dobrać odpowiednie wbudowane instrukcje, procedury, funkcje (metody) do określonych zadań;
	EE.09.1(7)4 wywołać instrukcje, funkcje (metody) i procedury;
EE.09.1(8) tworzy własne funkcje, procedury, obiekty, metody wybranych języków programowania;	EE.09.1(8)1 stworzyć własne obiekty;
	EE.09.1(8)2 przypisać wartości obiektom;
	EE.09.1(8)3 stworzyć własne procedury i funkcje (metody);
	EE.09.1(8)4 wywoływać własne procedury i funkcje (metody);
	EE.09.1(8)5 zanalizować poprawność tworzonych procedur, funkcji (metod) i obiektów;
EE.09.1(9) kompiluje i uruchamia kody źródłowe;	EE.09.1(9)1 scharakteryzować pojęcia kompilator, kod źródłowy;
	EE.09.1(9)2 skompilować lub uruchomić kod źródłowy;
EE.09.1(10) stosuje gotowe rozwiązania programistyczne;	EE.09.1(10)1 zastosować gotowe biblioteki podczas implementacji aplikacji;
	EE.09.1(10)2 zastosować gotowe algorytmy do rozwiązywania zadań programistycznych;
EE.09.1(11) testuje tworzoną aplikację i modyfikuje jej kod źródłowy;	EE.09.1(11)1 przeprowadzić testy aplikacji
	EE.09.1(11)2 zanalizować testy aplikacji
	EE.09.1(11)3 zmodyfikować kody źródłowe na podstawie analizy testów;
EE.09.1(12) dokumentuje tworzoną aplikację.	EE.09.1(12)1 zastosować komentarze i uwagi w kodzie źródłowym aplikacji
	EE.09.1(12)2 stworzyć pomoc do własnej aplikacji
	EE.09.1(12)3 stworzyć instrukcje do własnej aplikacji

Wybrane języki programowania w programowaniu obiektowym to C++ i C#. Wszystkie zapisy dotyczące treści kształcenia, efektów kształcenia, a także wymagań edukacyjnych dotyczą języków C++ i C#.

opracował

Marek Tarabasz