

Wymagania edukacyjne

Projektowanie oprogramowania/Pracownia projektowania oprogramowania Technik programista

Sposoby sprawdzania i oceniania osiągnięć edukacyjnych uczniów. Ocenie podlega zarówno wiedza teoretyczna, jak i nabyte w trakcie nauki umiejętności.

Oceniane są:

1. *Ćwiczenia, zadania wykonywane na lekcji*

Ocenie podlega: wykonanie wszystkich poleceń zgodnie z treścią, stopień samodzielności wykonywania zadania, pilność, końcowy efekt pracy (jakość pracy), umiejętność pracy w zespole, terminowość – warunek otrzymania oceny dobrej i wyższej.

2. *Aktywność podczas pracy na lekcji.*

Ocenie podlega: aktywność ucznia w czasie zajęć, stopień zaangażowania podczas wykonywania zajęć, zainteresowanie tematem lekcji.

3. *Kartkówki, testy, sprawdziany pisemne lub praktyczne.*

Ocena z prac pisemnych zgodna ze Statutem Szkoły

4. *Zadania dodatkowe, prace projektowe*

5. *Skala ocen:*

Celujący (6): 100%

Bardzo dobry (5): 90% – 99%

Dobry (4): 75% – 89%

Dostateczny (3): 51% – 74%

Dopuszczający (2): 30% – 50%

Niedostateczny (1): 0% – 29%

Wymagania na poszczególne oceny				
Dopuszczający	Dostateczny	Dobry	Bardzo dobry	Celujący
Przygotowanie środowiska pracy				
- Uruchamia edytor kodu (np. VS Code). - Tworzy plik .html i otwiera go w przeglądarce. - Otwiera kartę Konsola (F12) w przeglądarce.	- Instaluje wtyczki ułatwiające pracę (np. Live Server). - Używa debuggera przeglądarki do zatrzymywania kodu (Breakpoints).	- Tworzy lokalne repozytorium Git. - Wysyła kod (commit i push) na GitHub lub GitLab.	- Pracuje na gałęziach (branches) i używa pliku .gitignore. - Rozwiązuje konflikty kodu (merge, rebase).	- Konfiguruje automatyczne testowanie i wdrażanie aplikacji (CI/CD). - Uruchamia środowisko w kontenerach Docker.
Elementy programowania (JavaScript)				
- Osadza kod JavaScript w pliku HTML za pomocą znaczników <script>. - Tworzy zmienne i stałe (let, const). - Stosuje typy danych: string, number, boolean. - Wypisuje komunikaty w konsoli przez console.log(). - Znajduje błędy składniowe w kodzie na podstawie wskazań konsoli.	- Stosuje instrukcje warunkowe (if...else, switch). - Stosuje pętle (for, while) oraz tablice jednowymiarowe. - Tworzy funkcje z parametrami, które zwracają wartość (return). - Pobiera dane od użytkownika przez prompt() i alert(). - Formatuje napisy za pomocą szablonów tekstowych (` `). - Wyjaśnia różnicę między liczbą całkowitą a zmiennoprzecinkową.	- Piszze czytelny kod zgodnie z zasadami Clean Code. - Tworzy klasy (class), konstruktory i metody wewnątrz pliku HTML. - Stosuje metody tablicowe: .forEach(), .map(), .filter(). - Zabezpiecza kod przed awarią za pomocą bloków try...catch. - Używa funkcji console.table() do wyświetlania danych. - Koduje sortowanie o złożoności $O(n^2)$ (bąbelkowe lub przez wstawianie).	- Stosuje zaawansowane kolekcje danych: Map oraz Set. - Koduje algorytmy typu „dziel i zwyciężaj” (wyszukiwanie binarne oraz Quick Sort lub Merge Sort).	- Pobiera dane z zewnętrznych serwerów przy użyciu funkcji fetch(). - Zapisuje i odczytuje dane strukturalne JSON w localStorage / sessionStorage. - Świadomie zarządza kontekstem this, funkcjami strzałkowymi i domknięciami (Closures). - Optymalizuje algorytmy sortowania pod kątem oszczędności pamięci (in-place).

	- Koduje szyfr przesuwający ROT (np. ROT13).			
Projektowanie aplikacji				
- Wskazuje w gotowym kodzie zmienne, funkcje, klasy i obiekty. - Odczytuje proste, gotowe diagramy klas oraz przypadków użycia UML.	- Wyjaśnia różnicę między programowaniem strukturalnym a obiektowym. - Rysuje proste schematy blokowe dla algorytmów (np. dla szyfru ROT). - Wyjaśnia różnicę między kaskadowym (Waterfall) a zwinnym (Agile) modelem pracy.	- Stosuje w projektach cztery filary programowania obiektowego (hermetyzacja, dziedziczenie, polimorfizm, abstrakcja). - Tworzy kompletną dokumentację UML dla projektowanego systemu. - Opisuje role i etapy pracy w zespole według metodyki Scrum. - Rysuje schemat blokowy przedstawiający proces zamiany elementów w tablicy podczas sortowania.	- Zna i dobiera podstawowe wzorce projektowe (np. Singleton, Fabryka, MVC). - Projektuje architekturę aplikacji wielowarstwowych (przepływ danych między modułami). - Wyjaśnia teoretyczne zasady działania strategii „dziel i zwyciężaj”.	- Projektuje od podstaw autorskie, skalowalne systemy oprogramowania dla realnych problemów. - Adaptuje technikę „dziel i zwyciężaj” do rozwiązywania niestandardowych problemów optymalizacyjnych.