

Wymagania edukacyjne

Projektowanie oprogramowania

Klasa I i klasa II – technik programista

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

Oceniane są:

- 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.
- Aktywność podczas pracy na lekcji.
Ocenie podlega: aktywność ucznia w czasie zajęć, stopień zaangażowania podczas wykonywania zajęć, zainteresowanie tematem lekcji.
- Kartkówki, sprawdziany pisemne lub praktyczne.
Ocena z prac pisemnych zgodna ze Statutem Szkoły.
- Zadania dodatkowe, prace projektowe.

Wymagania na poszczególne oceny				
dopuszczający	dostateczny	dobry	bardzo dobry	celujący
Zasady projektowanie oprogramowania				
Zna podstawowe pojęcia związane z projektowaniem oprogramowania, takie	Umie posługiwać się prostymi typami danych oraz rozróżnia typy	Projektuje algorytmy za pomocą schematów blokowych, list kroków i	Samodzielnie projektuje aplikację opartą na architekturze klient-serwer, uwzględniając	Tworzy pełny projekt aplikacji, w tym dokumentację techniczną, interfejs

<p>jak: typy danych, algorytm, interfejs użytkownika.</p> <p>Umie korzystać z prostych narzędzi do projektowania, takich jak edytor tekstowy i schematy blokowe.</p> <p>Rozróżnia podstawowe paradygmaty projektowania (np. strukturalne i obiektowe).</p> <p>Potrafi zastosować podstawowe algorytmy sortowania i wyszukiwania.</p> <p>Zna zasady bezpieczeństwa danych i potrafi wymienić podstawowe zagrożenia w cyberprzestrzeni.</p>	<p>złożone (tablice, rekordy).</p> <p>Potrafi zaprojektować algorytm dla prostego problemu i zapisać go w pseudokodzie lub jako schemat blokowy.</p> <p>Rozumie, jak korzystać z narzędzi do zarządzania projektami, takich jak diagramy Gantta.</p> <p>Potrafi zaprojektować prosty interfejs użytkownika i opisać jego funkcje.</p> <p>Zna podstawowe zasady ochrony danych osobowych i przestrzega zasad prywatności w cyfrowym świecie.</p>	<p>pseudokodu, analizując ich złożoność.</p> <p>Potrafi zastosować złożone typy danych (kolekcje, struktury, wskaźniki) w projektach programistycznych.</p> <p>Umie zaprojektować interfejs użytkownika dostosowany do różnych platform.</p> <p>Stosuje zasady projektowania aplikacji, w tym hermetyzację, dziedziczenie i polimorfizm.</p> <p>Korzysta z narzędzi wspierających zarządzanie projektami, takich jak Jira, Trello, oraz systemów kontroli wersji (Git).</p>	<p>analizę wymagań klienta.</p> <p>Stosuje różne wzorce projektowe (np. Fasada, Kompozyt) i potrafi dobrać odpowiedni wzorzec do zadania.</p> <p>Projektuje zaawansowane algorytmy (rekurencyjne, heurystyczne) i analizuje ich złożoność.</p> <p>Umie planować i organizować prace projektowe, stosując metodyki zwinne (Scrum, Kanban).</p> <p>Zapewnia zgodność projektu z normami i procedurami oceny jakości.</p>	<p>użytkownika, funkcjonalność i zabezpieczenia.</p> <p>Stosuje zaawansowane wzorce projektowe i wykorzystuje dokumentację techniczną do rozwiązywania problemów.</p> <p>Rozwiązuje problemy optymalizacyjne w aplikacjach, korzystając z algorytmów i analiz złożoności.</p> <p>Projektuje aplikacje w różnych paradygmatach, takich jak strukturalny, obiektowy, i klient-serwer.</p> <p>Inicjuje i prowadzi prace zespołowe nad projektami programistycznymi, efektywnie organizując zasoby.</p>
---	---	---	--	---