Technikum w Zespole Szkół
im. Armii Krajowej
Obwodu “Głuszec” - Grójec
w Grójcu

Wymagania edukacyjne
na poszczególne oceny szkolne z przedmiotu:
Pracownia Podstaw Programowania

I. Podstawa prawna

1. Ustawa z dnia 7 września 1991 r. o systemie oświaty (tekst jednolity: Dz.U. z 2024 r., poz. 750) - Rozdział 3a
2. Ustawa z dnia 14 grudnia 2016 r. Prawo oświatowe (Dz.U.2023 poz.900)
3. Rozporządzenie Ministra Edukacji Narodowej z dnia 22 lutego 2019 r. w sprawie oceniania, klasyfikowana i promowania uczniów i słuchaczy w szkołach publicznych (tekst jedn.: Dz.U. z 2023 r., poz. 2572)
4. Statut Technikum w Zespole Szkół im. Armii Krajowej Obwodu “Głuszec” - Grójec w Grójcu.
5. Program nauczania dla zawodu Technik Programista 351406

II. Wymagania edukacyjne na poszczególne oceny szkolne:

INF.04.3. Pracownia podstaw programowania	
Efekty kształcenia	Kryteria weryfikacji
Uczeń	Uczeń
1) posługuje się prostymi typami danych	1) rozróżnia typy numeryczne, stałoprzecinkowe i zmiennoprzecinkowe 2) rozpoznaje typ logiczny 3) rozróżnia typy znakowe i łańcuchowe 4) posługuje się typami liczbowymi stałoprzecinkowymi i zmiennoprzecinkowym, typem logicznym, typem znakowymi i łańcuchowym
2) posługuje się złożonymi typami danych	1) rozróżnia rodzaje złożonych typów danych 2) posługuje się tablicami jednowymiarowymi i dwuwymiarowymi 3) posługuje się tablicami asocjacyjnymi i dynamicznymi 4) posługuje się typem rekordowym 5) posługuje się typem plikowym 6) posługuje się typem wskaźnikowym 7) charakteryzuje cechy kolekcji, w tym znaczenie iteratora 8) posługuje się kolekcjami, np. kolejkami, listami, stosami, wektorami 9) projektuje zestawy danych dla problemu programistycznego

3) stosuje metody rozwiązywania problemów za pomocą algorytmów	1) projektuje algorytmy za pomocą różnych metod: schematów blokowych listy kroków, drzew decyzyjnych, pseudokodu 2) charakteryzuje algorytmy iteracyjne, tekstowe i szyfrowania, tablicowe 3) charakteryzuje algorytmy rekurencyjne 4) charakteryzuje problemy i metody ich rozwiązania, np. algorytmy heurystyczne, problem komiwojażera 5) określa złożoność obliczeniową algorytmów
4) stosuje algorytmy sortowania i wyszukiwania	1) charakteryzuje typy sortowania i ich złożoność obliczeniową 2) stosuje różne typy sortowania, np. bąbelkowe, zachłanne, przez wstawianie, szybkie, metodą dziel i zwyciężaj 3) stosuje algorytmy wyszukiwania dla tablic, list, kolejek i stosów
5) stosuje wzorce projektowe	1) dobiera wzorzec projektowy do zadania programistycznego 2) stosuje wzorce projektowe w programowaniu obiektowym, np. Metoda szablonowa (Template method), Fasada (Facade), Kompozyt (Composite)
6) stosuje zagadnienia prawa autorskiego w dziedzinie programowania	1) rozróżnia autorskie prawa osobiste i majątkowe 2) określa czas trwania praw autorskich 3) określa konsekwencje naruszenia prawa autorskiego 4) charakteryzuje elementy własności intelektualnej 5) rozróżnia typy licencji oprogramowania

Ocena niedostateczna (1)

- Uczeń nie potrafi rozpoznać prostych i złożonych typów danych ani nie zna metod rozwiązywania problemów za pomocą algorytmów.
- Uczeń nie rozumie podstawowych typów danych, takich jak typ logiczny, liczbowy czy znakowy.
- Uczeń nie potrafi napisać programu, który korzysta z prostych ani złożonych typów danych oraz nie stosuje algorytmów sortowania ani wzorców projektowych.
- Uczeń nie potrafi przeanalizować problemu programistycznego i zaprojektować algorytmu.

- Uczeń nie potrafi zaplanować rozwiązań problemów za pomocą algorytmów ani zaprojektować zestawów danych do programów.
- Uczeń nie ocenia poprawności napisanych programów i nie przestrzega prawa autorskiego.

Ocena dopuszczająca (2)

- Uczeń rozróżnia podstawowe typy danych (numeryczne, logiczne, znakowe) i zna podstawowe algorytmy, takie jak sortowanie bąbelkowe.
- Uczeń rozumie podstawowe różnice między typami danych prostych i złożonych oraz potrafi określić cel używania algorytmów.
- Uczeń potrafi napisać prosty program, który wykorzystuje podstawowe typy danych (np. tablice jednowymiarowe) i używa podstawowych algorytmów sortowania.
- Uczeń potrafi analizować proste problemy programistyczne, identyfikować dane potrzebne do ich rozwiązania oraz zastosować podstawowe struktury danych (np. tablice).
- Uczeń podejmuje próby stworzenia prostych algorytmów, ale wymaga pomocy w organizowaniu danych i rozwiązywaniu bardziej złożonych problemów.
- Uczeń rozpoznaje podstawowe zasady prawa autorskiego, ale ma trudności z ich praktycznym stosowaniem.

Ocena dostateczna (3)

- Uczeń zna i potrafi rozróżnić wszystkie proste typy danych oraz podstawowe złożone typy danych, takie jak tablice dwuwymiarowe, kolekcje (np. listy, kolejki) i rekordy.
- Uczeń rozumie działanie algorytmów sortowania i wyszukiwania oraz zna podstawowe wzorce projektowe (np. Fasada, Kompozyt).
- Uczeń potrafi napisać programy z wykorzystaniem tablic, kolekcji oraz prostych algorytmów iteracyjnych, sortowania i wyszukiwania.
- Uczeń potrafi analizować i rozwiązywać typowe problemy programistyczne, takie jak sortowanie danych czy wyszukiwanie elementów w tablicach i listach.
- Uczeń potrafi zaprojektować algorytmy za pomocą pseudokodu lub schematów blokowych, a także zaplanować zestawy danych dla prostych problemów.
- Uczeń potrafi ocenić wydajność podstawowych algorytmów i rozpoznaje różne typy licencji oprogramowania oraz podstawowe zasady ochrony własności intelektualnej.

Ocena dobra (4)

- Uczeń dobrze zna i potrafi zastosować różne złożone typy danych, takie jak tablice asocjacyjne, dynamiczne i wskaźniki oraz rozumie zasady działania algorytmów rekurencyjnych.
- Uczeń dobrze rozumie złożoność obliczeniową algorytmów sortowania i wyszukiwania oraz zna zasady stosowania wzorców projektowych w programowaniu obiektowym.

- Uczeń potrafi stosować złożone struktury danych, takie jak rekordy i kolekcje (np. stosy, kolejki, wektory), oraz implementować różne algorytmy sortowania i wyszukiwania w praktyce.
- Uczeń potrafi analizować skomplikowane problemy, takie jak problem komiwojażera, oraz charakteryzować różne algorytmy rozwiązywania problemów (np. algorytmy heurystyczne).
- Uczeń potrafi zaprojektować i zaimplementować bardziej złożone algorytmy oraz dobrać odpowiednie struktury danych do problemów programistycznych.
- Uczeń potrafi ocenić efektywność zastosowanych algorytmów oraz rozpoznaje autorskie prawa majątkowe i osobiste, potrafi wskazać konsekwencje ich naruszenia.

Ocena bardzo dobra (5)

- Uczeń posiada bardzo dobrą znajomość różnych typów danych i algorytmów, w tym algorytmów rekurencyjnych, iteracyjnych, tekstowych i szyfrowania.
- Uczeń bardzo dobrze rozumie złożoność obliczeniową algorytmów oraz zasady stosowania wzorców projektowych w programowaniu obiektowym (np. Template Method, Kompozyt).
- Uczeń potrafi zaprojektować i zaimplementować wydajne rozwiązania programistyczne, stosując odpowiednie typy danych, algorytmy i wzorce projektowe w złożonych programach.
- Uczeń potrafi przeprowadzić szczegółową analizę problemu programistycznego i dobrać odpowiednie algorytmy oraz struktury danych do jego rozwiązania.
- Uczeń potrafi stworzyć kompleksowy projekt programistyczny, łącząc różne algorytmy i wzorce projektowe, dbając o wydajność i skalowalność programu.
- Uczeń potrafi ocenić jakość i efektywność własnych rozwiązań, stosuje przepisy prawa autorskiego oraz potrafi rozpoznać różne typy licencji oprogramowania.

Ocena celująca (6)

- Uczeń posiada wszechstronną wiedzę na temat wszystkich typów danych i zaawansowanych algorytmów, w tym algorytmów heurystycznych i problemów NP-trudnych.
- Uczeń doskonale rozumie złożoność obliczeniową skomplikowanych algorytmów oraz zasady tworzenia zaawansowanych struktur danych i wzorców projektowych.
- Uczeń potrafi samodzielnie tworzyć innowacyjne rozwiązania, korzystając z zaawansowanych typów danych i algorytmów, tworząc wydajne i skalowalne aplikacje.
- Uczeń potrafi przeprowadzać szczegółową analizę problemów programistycznych i optymalizować algorytmy pod kątem wydajności oraz złożoności obliczeniowej.
- Uczeń tworzy złożone systemy programistyczne, integrując różne typy danych, zaawansowane algorytmy oraz wzorce projektowe w sposób optymalny.
- Uczeń potrafi krytycznie ocenić i udoskonalić swoje projekty, przestrzegając przepisów prawa autorskiego i stosując odpowiednie licencje oprogramowania, dbając o jakość kodu oraz etykę pracy programistycznej.