

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

Wymagania edukacyjne na poszczególne oceny  
szkolne z przedmiotu:

Programowanie obiektowe

## 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, klasyfikowania 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 "Głuszczyk" - Grójec w Grójcu.
5. Program nauczania dla zawodu Technik Programista 351406

Efekty kształcenia z podstawy programowej	Kryteria weryfikacji z podstawy programowej
Uczeń:	Uczeń:
- Wykorzystuje środowisko programistyczne dla obiektowych aplikacji konsolowych.	- Rozróżnia kompilatory i interpretery. - Charakteryzuje zadania kompilatora, interpretera, debugera. - Analizuje błędy w kodzie za pomocą debugera. - Charakteryzuje etapy kompilacji i interpretacji kodu. - Charakteryzuje pojęcie biblioteki. - Kompiluje i uruchamia programy. - Objaśnia pojęcia kompilator, interpreter oraz zasady ich działania.
- Przestrzega zasad programowania.	- Dzieli program na funkcje (metody). - Stosuje rekurencję. - Implementuje algorytmy w programie.
- Korzysta z typów danych.	- Stosuje proste i złożone typy danych - Deklaruje własne typy danych - Deklaruje zmienne różnych typów - Wykonuje operacje na zmiennych: wejścia-wyjścia, arytmetyczne, logiczne - Stosuje typy złożone i operacje na nich
- Stosuje wyrażenia, instrukcje i biblioteki.	- Stosuje operatory arytmetyczne, przypisania, porównania, logiczne, bitowe, operatory do obsługi łańcuchów - Wykorzystuje priorytety operatorów do właściwego budowania wyrażeń

	<ul style="list-style-type: none"> <li>- Stosuje instrukcję warunkową i wyboru</li> <li>- Stosuje instrukcje pętli</li> <li>- Korzysta z wybranych bibliotek języka C++/lub C# lub Java lub Python lub innego popularnego języka programowania: biblioteki standardowej, biblioteki z funkcjami matematycznymi, biblioteki z podstawowymi algorytmami</li> </ul>
- Stosuje zasady programowania obiektowego	<ul style="list-style-type: none"> <li>- Stosuje obiektowe podejście do rozwiązywania problemów</li> <li>- Charakteryzuje pojęcia klasa, obiekt, metoda, pole, dziedziczenie, hermetyzacja, polimorfizm</li> <li>- Dzieli zagadnienie na klasy</li> <li>- Powołuje obiekty</li> <li>- Planuje aplikację z zastosowaniem hermetyzacji, dziedziczenia i polimorfizmu</li> </ul>
- Definiuje klasy	<ul style="list-style-type: none"> <li>- Definiuje pola klasy</li> <li>- Określa zakres widzialności pól klasy i definiuje kwalifikatory dostępu</li> <li>- Definiuje metody klasy</li> <li>- Definiuje konstruktory (w tym kopiujący) i destruktor klasy</li> <li>- Definiuje instrukcje inicjujące konstruktora</li> <li>- Określa zakres widzialności metod klasy i definiuje kwalifikatory dostępu</li> <li>- Implementuje funkcjonalność klasy</li> <li>- Deklaruje obiekty i odwołuje się obiektem do składowych klasy</li> <li>- Definiuje składniki statyczne klasy</li> <li>- Rozróżnia klasy dziedziczone i zaprzyjaźnione</li> <li>- Tworzy funkcje zaprzyjaźnione z klasą, stosuje składnik statyczny klasy i metody do ich obsługi</li> </ul>
- Definiuje klasy pochodne	<ul style="list-style-type: none"> <li>- Buduje hierarchię dziedziczenia klas w programie</li> <li>- Wydziela metody i pola do odpowiednich klas w hierarchii dziedziczenia</li> <li>- Definiuje klasy bazowe i pochodne</li> <li>- Stosuje metody wirtualne, definiuje klasy abstrakcyjne</li> </ul>
- Programuje szablony (wzorce) klas	<ul style="list-style-type: none"> <li>- Definiuje szablony klas do obsługi prostych typów liczbowych</li> </ul>
- Programuje obsługę wyjątków	<ul style="list-style-type: none"> <li>- Stosuje szkielet obsługi wyjątków z instrukcjami try i catch</li> <li>- Stosuje instrukcję throw</li> <li>- Opracowuje listę możliwych błędów wykonania aplikacji</li> <li>- Definiuje obsługę błędów wykonania aplikacji</li> </ul>

**Ocenę niedostateczną** otrzymuje uczeń, który:

- 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 częściowe oceny niedostateczne, których nie poprawia,
- Nie uczestniczy w projektach zespołowych.

**Ocenę dopuszczającą** otrzymuje uczeń, który:

Potrafi zdefiniować podstawowe pojęcia związane z programowaniem obiektowym, takie jak: Klasa, Obiekt, Dziedziczenie, Polimorfizm, Enkapsulacja.

Potrafi opisać podstawowe zasady działania programowania obiektowego, w tym: Jak tworzy się klasy i obiekty, Jak działa dziedziczenie, Jakie są korzyści z używania polimorfizmu, Jakie są zasady enkapsulacji

Potrafi podać przykłady prostych klas i obiektów, np.: Klasa Samochód z atrybutami marka, model, rok oraz Obiekt mojSamochod utworzony na podstawie klasy Samochód.

Potrafi napisać prosty program, który Tworzy klasę i obiekt oraz Ustawia i odczytuje wartości atrybutów obiektu

Potrafi rozpoznać i nazwać różne elementy programu obiektowego, takie jak: Klasy, Obiekty, Metody, Atrybuty

Wykonuje ćwiczenia z pomocą nauczyciela lub kolegi z zespołu

Zna i stosuje funkcje wejścia/wyjścia,

Potrafi zadeklarować podstawowe zmienne prostego typu

**Ocenę dostateczną** otrzymuje uczeń, który:

Potrafi napisać prosty program w języku obiektowym, który: Tworzy klasę i obiekt, Ustawia i odczytuje wartości atrybutów obiektu, Definiuje i wywołuje metody klasy.

Potrafi wyjaśnić funkcje poszczególnych elementów programu obiektowego i ich rolę w kodzie.

Potrafi napisać program, który tworzy obiekt i wywołuje jego metody, np.: Klasa Samochód z metodą jedz(), która wypisuje komunikat "Samochód jedzie".

Zna strukturę, składnię, typy danych, operatory, posługuje się zmiennymi, zna instrukcje iteracyjne i warunkowe i potrafi ich używać

Definiuje funkcje, zna pojęcie zasięgu zmiennych, zna wybrane funkcje z biblioteki standardowej

Dzieli program na klasy i funkcje.

**Ocenę dobrą** otrzymuje uczeń, który:

Potrafi analizować kod źródłowy, identyfikować błędy oraz proponować potencjalne usprawnienia.

Potrafi przeprowadzić refaktoryzację kodu w celu poprawy jego czytelności i efektywności.

Potrafi ocenić efektywność i złożoność obliczeniową różnych algorytmów i struktur danych w kontekście programowania obiektowego.

Potrafi porównać różne podejścia do rozwiązania problemu i wybrać najbardziej optymalne.

Potrafi zidentyfikować i naprawić błędy w kodzie, np.: błędy logiczne w metodach klasy, oraz błędy związane z dziedziczeniem i polimorfizmem.

Potrafi zaproponować ulepszenia w istniejącym kodzie, np.: Zoptymalizować algorytmy pod kątem wydajności oraz Zastosować bardziej efektywne struktury danych.

Samodzielnie wykonuje ćwiczenia i rozwiązania mają co najwyżej drobne niedoskonałości.

Dobiera odpowiednie algorytmy i struktury danych

Testuje stworzone programy.

**Ocenę bardzo dobrą** otrzymuje uczeń, który:

Potrafi samodzielnie zaprojektować złożony program obiektowy, uwzględniając zaawansowane techniki, takie jak: Dziedziczenie, Polimorfizm, Enkapsulacja.

Potrafi zaimplementować zaprojektowany program, dbając o czytelność i efektywność kodu.

Potrafi krytycznie ocenić własne projekty oraz projekty innych, proponując konkretne usprawnienia.

Potrafi przeprowadzić kompleksową analizę złożoności obliczeniowej i efektywności zaprojektowanych algorytmów.

Potrafi optymalizować kod pod kątem wydajności i złożoności obliczeniowej.

Potrafi stworzyć aplikację z wieloma klasami, które dziedziczą po sobie i współpracują ze sobą, np: System zarządzania biblioteką, gdzie klasy reprezentują książki, czytelników, wypożyczenia itp.

Potrafi zastosować zaawansowane techniki programowania obiektowego, takie jak: Wzorce projektowe (np. Singleton, Factory, Observer) oraz Testowanie jednostkowe.

Oddaje prace niemal bez wad, zawierające niewpływające na jakość pracy błędy lub niedociągnięcia

Wyznacza złożoność obliczeniową i pamięciową wybranych algorytmów i ocenia ich efektywność

Testuje stworzone programy dla przypadków typowych, danych błędnych oraz brzegowych.

**Ocenę celującą** otrzymuje uczeń, który:

Potrafi tworzyć innowacyjne rozwiązania programistyczne, wykorzystując zaawansowane wzorce projektowe i techniki optymalizacji.

Potrafi zaprojektować i zaimplementować kompleksowy system, który integruje różne technologie i narzędzia programistyczne.

Potrafi zaprojektować architekturę oprogramowania, uwzględniając wzorce projektowe, takie jak MVC (Model-View-Controller), Singleton, Factory, Observer, itp.

Uczeń potrafi stworzyć zaawansowaną aplikację, która wykorzystuje różne wzorce projektowe i techniki optymalizacji, np.: System zarządzania przedsiębiorstwem, który integruje moduły finansowe, kadrowe, magazynowe itp.

- Bierze udział w projektach zespołowych jako odpowiedzialny lider projektu
- Perfekcyjnie, samodzielnie i w wyznaczonym, czasie wykonuje ćwiczenia i powierzone zadania.
- Pomysłowo rozwiązuje zadania nietypowe lub o wyższym stopniu trudności, np. z olimpiady informatycznej