Nauka programowania w Moodle

Piotr Szymczyk
e-mail: pszymczyk@wsei.edu.pl

STRESZCZENIE Artykuł opisuje możliwości nauczania programowania przy wykorzystaniu wirtualnego laboratorium programowania oraz pytań typu CodeRunner w testach w pakiecie Moodle. Rozwiązania te dają dodatkowe możliwości dydaktyczne, zwiększają poziom nauczania, ćwiczą praktyczną stronę programowania. Dodatkową wartością jest automatyzacja żmudnych czynności procesu nauczania takich jak weryfikacja poprawności kodu, poprawianie błędów oraz ocena kodu źródłowego na zajęciach laboratoryjnych czy podczas sprawdzianów nabytej wiedzy.

SŁOWA KLUCZOWE nauka programowania, Moodle, VPL, wirtualne laboratorium programowania, CodeRunner, automatyczne testy kodu, automatyczne ocenianie kodu

1 Wprowadzenie

Proces nauki programowania jest dość złożony, długotrwały i wieloetapowy. Wiedza teoretyczna musi zostać ugruntowana poprzez praktyczne ćwiczenia na zajęciach komputerowych w laboratorium. Wykorzystanie narzędzi dostępnych w Moodle daje wiele korzyści jak i możliwości. W dalszej części artykułu zostaną opisane dwie możliwości jakie daje Moodle – wirtualne laboratorium programowania oraz pytania testowe typy CodeRunner.

2 Zasadniczy opis

2.1 Wirtualne laboratorium programowania

Wirtualne laboratorium programowania (VPL - Virtual Programming Lab) [1] jest jedną z wielu aktywności systemu Moodle. Zostało ono stworzone na Uniwersytecie de Las Palmas de Gran Canaria. Jest to tak zwany plugin do Moodle [2], dostępny za darmo ze wsparciem społeczności użytkowników oraz twórców.

W wirtualnym laboratorium możemy zdefiniować i skonfigurować pracę studentów nad programem komputerowym w ramach laboratorium lub projektu. Można utworzyć testy jednostkowe oraz szczegółowe wymagania co do treści programu (konieczność użycia na przykład jakiś konstrukcji językowych lub ich wykluczenie). Student ma dostępny prosty edytor tekstowy, w którym tworzy kod źródłowy programu, kompilator danego języka oraz debugger. Po poprawnym skompilowaniu program może być poddany automatycznym testom i zależnie od wyniku tych testów jest automatycznie oceniany. Prowadzący może w każdej chwili podglądać pracę studenta nad kodem źródłowym i w razie potrzeby skonsultować problemy oraz zobaczyć zapisane w logu postępy nad kodem programu w skali czasu, co pozwala na wyłapanie nieprawidłowości, na przykład wkopiowane do edytora całego programu w jednej chwili czy też przepisywanie kodu.

Tabela 1 Języki programowania wspierane przez wirtualne laboratorium programowania

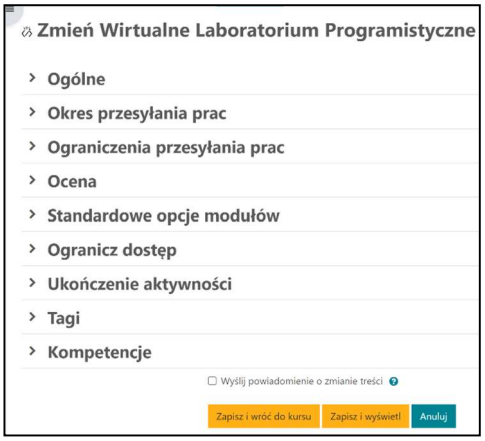
Ada	Assembler x86	C	Clojure	C++	D	Erlang	Fortran	Go
Groovy	Haskell	HTML	Java	JavaScript	Kotlin	Lisp	Lua	Octave
Matlab	MiniZinc	MIPS	Pascal	Perl	PHP	Prolog	Python	R
Ruby	Scala	Scheme	Shell	SQL	TypeScript	Verilog	VHDL	

W tabeli 1 pokazano wspierane języki programowania, w ramach języka programowania jest też często możliwość wyboru wersji języka, na przykład dla C++ mamy obecnie (lipiec 2023) do wyboru: C++98, C++11, C++14 lub C++17, a dla języka C: ANSI-C, C99 lub C11.

Aby skorzystać z VPL należy dodać tą aktywność do kursu. Po jej dodaniu na stronie kursu pojawi się odpowiednia ikona, tak jak to pokazano na Rys.1.

Rysunek 1 Ikona wirtualnego laboratorium programowania

Następnie należy skonfigurować parametry wirtualnego laboratorium podobnie jak dla innych aktywności w Moodle (Rys.2).



Rysunek 2 Konfiguracja ogólnych parametrów aktywności VPL

Aby wykonać tą operację należy być w trybie edycyjnym kursu i dla aktywności VPL wybrać z jej prawej strony opcje ikonę trzech pionowych kropek, a następnie „***Edytuj ustawienia***”.

W części „***Ogólne***” możemy określić:

- Nazwę
- Krótki opis
- Pełny opis
- Czy pełny opis będzie wyświetlany na stronie kursu

Ważne jest prawidłowe wypełnienie tych informacji ze względu na to, że mając w kursie kilka aktywności VPL, a przeważnie jest ich tyle ile tematów ćwiczeń, łatwo będzie można wtedy zidentyfikować, której dotyczą poszczególne oceny.

W części „***Okres przesyłania prac***” podajemy:

- Od kiedy będzie dostępna aktywność
- Data końca dostępności

Obydwa parametry można włączyć lub wyłączyć oraz podać dokładny czas (dzień, godzina, minuty). Warto tę możliwości wykorzystać aby studenci mieli dostęp do ćwiczenia tylko w trakcie zaplanowanych zajęć lub do końca semestru itp.

W części „***Ograniczenia przesyłania prac***” określamy:

- Maksymalną liczbę plików – zależnie od specyficznych potrzeb danego laboratorium, zwykle jest to jeden plik
- Rodzaj pracy – czy indywidualna czy grupowa, zwykle jest to praca indywidualna
- Czy dopuszczamy przesyłanie zewnętrznego pliku
- Czy będą ćwiczenia przykładowe
- Jaki jest maksymalny rozmiar wysyłanego pliku
- Można też ustawić hasło dostępu
- Czy zezwalamy na przesyłanie z sieci
- Czy wymagana jest specjalna przeglądarka SEB
- Klucz do SEB

Użycie przeglądarka SEB (Safe Exam Browser) [3] umożliwia blokowanie innych aplikacji i pełną kontrolę nad komputerem, z którego student ma dostęp do wirtualnego laboratorium. Przeglądarka ta skutecznie i całkowicie blokuje inne programy uruchomione na komputerze i przejmuje pełna kontrolę nad monitorami, klawiaturą i myszą. Użytkownik może wykonywać na komputerze tylko te czynności, które zostały dopuszczone w jej konfiguracji.

W części „***Ocena***” możemy określić:

- Ocena definiuje sposób oceniania tej aktywności, przeważnie są to punkty
- Maksymalna ocena, na przykład 100 punktów

- Kategoria ocen określa, w której kategorii dziennika jest umieszczana ocena
- Próg zaliczenia
- Czy ma być automatycznie zmniejszana ocena za każde automatyczne podejście do oceny
- Czy są darmowe oceny, które nie powodują zmniejszania końcowego wyniku
- Czy ocena jest widoczna

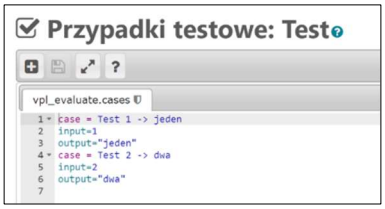
Warto również skorzystać z opcji „***Ogranicz dostęp***” aby dodatkowo ograniczyć dostęp do VPL na przykład w danym czasie tylko dla danej grupy studentów. Pozostałe części, „***Standardowe opcje modułów***”, „***Tagi***” oraz „***Kompetencje***” są identyczne jak dla innych aktywności Moodle.
W tak ogólnie skonfigurowana aktywności VPL należy następnie szczegółowo określić zadanie programistyczne oraz poszczególne parametry i potrzebne skrypty do kompilacji, uruchamiania, testowania i oceniania pracy.
Aby wykonać te operacje należy wyłączyć tryb edycji kursu i wejść do tej aktywności, wtedy ukażą się potrzebne zakładki.

Przypadki testowe

W tym miejscu możemy określić reguły sprawdzania poprawności programu poprzez podanie danych wejściowych i spodziewanych wyników w postaci danych wyjściowych. W tym celu należy przy przygotować zbiór o nazwie **vpl_evaluate.cases**, który ma ściśle określony format:

- case = Opis przypadku testowego
- input = text
- output = text
- grade reduction = [value | percentage %]

Szczegółowy opis tego formatu znajduje się w opcji Pomoc w Moodle oraz na stronie z dokumentacją VPL [4].
Przykładowa treść tego pliku pokazano na Rys.3.

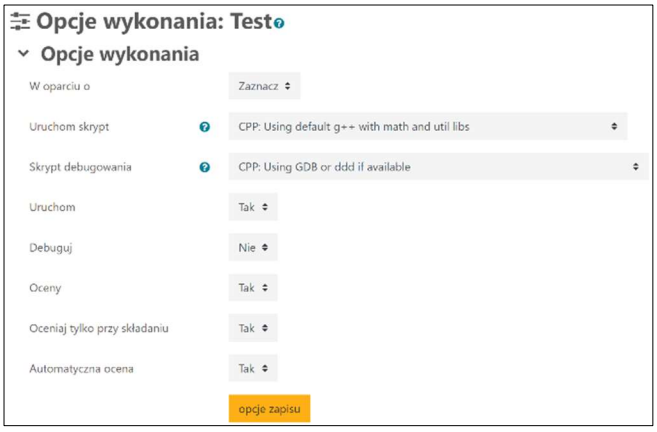


Rysunek 3 Przykładowa treść pliku vpl_evaluate.cases

Zdefiniowano tu dwa testy. Pierwszy test sprawdza czy po wczytaniu wartości liczbowej **1** program wypisuje na wyjściu tekst: „**jeden**”, a drugi czy po wczytaniu wartości **2** program wypisuje na wyjściu tekst „**dwa**”. Jeśli obydwa testy wykonają się prawidłowo to ocena nie będzie obniżana.

Opcje wykonania

W tym miejscu określamy parametry wykonania, między innymi jaki język programowania będzie użyty oraz jaką jego wersję wybieramy (Rys.4).



Rysunek 4 Opcje wykonania

Wymagane pliki

Dobrze jest zdefiniować nazwy (ważne są rozszerzenia) i wstępną treść plików kodu źródłowego, który będzie wyświetlany na początku pracy nad danym problemem wszystkim studentom w oknie edytora kodu źródłowego (Rys.5).

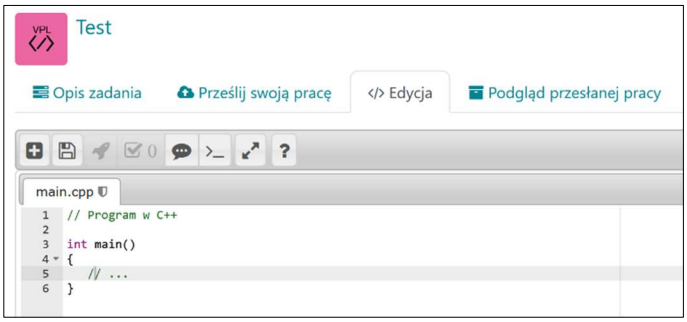
Rysunek 5 Wymagane pliki

Pozostałe opcje znajdują się w poszczególnych pozycja menu dostępne po rozwinięciu opcji „**Więcej**” i są to:

- Filtry
- Uprawnienia
- Kopia zapasowa
- Odtwórz
- Pliki wykonywalne
- Maksymalne limity zasobów wykonania
- Pliki używane podczas wykonania
- Wariacje
- Overrides
- Sprawdź serwery wykonawcze
- Lokalne serwery wykonawcze
- Prześlij swoją pracę
- Edycja
- Poprzednia lista składania
- Wirtualne Laboratoria Programowania

Szczegółowy ich opis można znaleźć się w dokumentacji, kontekstowej pomocy Moodle oraz przykładach dostępnych na stronie VPL demo [5].

Student po wejściu do danego wirtualnego laboratorium powinien przeczytać informacje znajdujące się w zakładce „Opis zadania” i przejść do zakładki „</> Edycja” gdzie zobaczy przygotowany przez prowadzącego plik kodu programu, który następnie powinien edytować i zapisać swój kod programu realizujący zadany temat laboratorium (Rys.6).

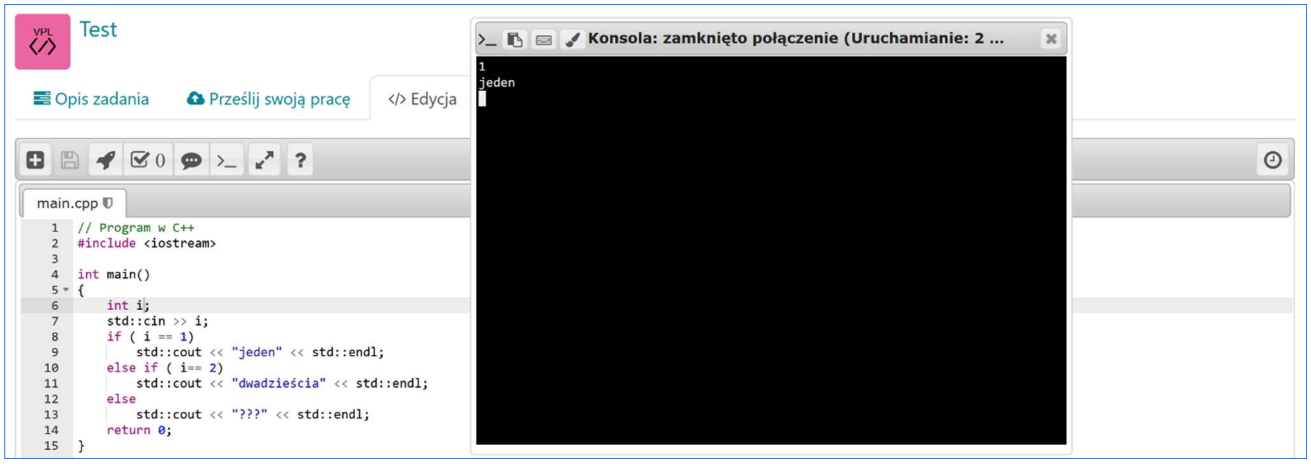


Rysunek 6 Zakładka Edycja

Po wykonaniu tej pracy i zapisaniu jej (ikona dyskietki) uaktywnia się opcja kompilacji (ikona rakiety) umożliwiająca skompilowanie programu.

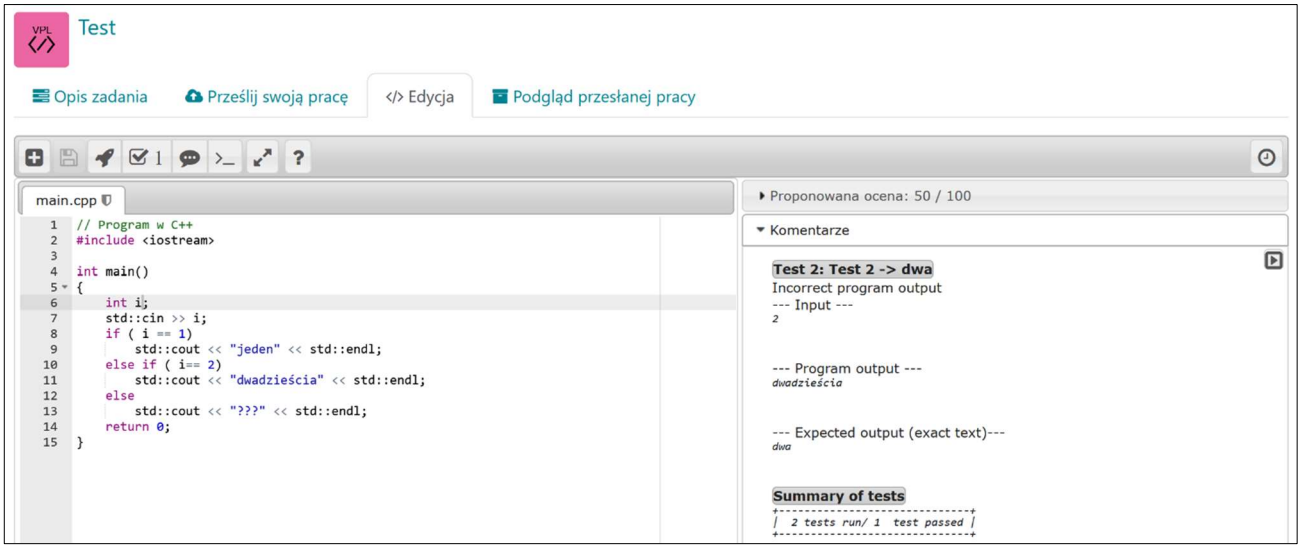
Rysunek 7 Kompilacja programu i analiza ewentualnych błędów kompilacji

Po próbie kompilacji otrzymujemy komunikaty ewentualnych błędów na Rys.7 pokazano z prawej strony wykryty błąd, który należy poprawić. Na podstawie komunikatu błędu oraz numeru linii łatwo jest się zorientować co należy skorygować w programie aby nie było błędów i była możliwość uruchomienia programu. Jeśli kompilacja przebiegnie bez błędów to uruchamiany jest program w trybie tekstowym (dla programów w C++) i oczekuje danych oraz będzie wypisywał wyniki jak na Rys.8.

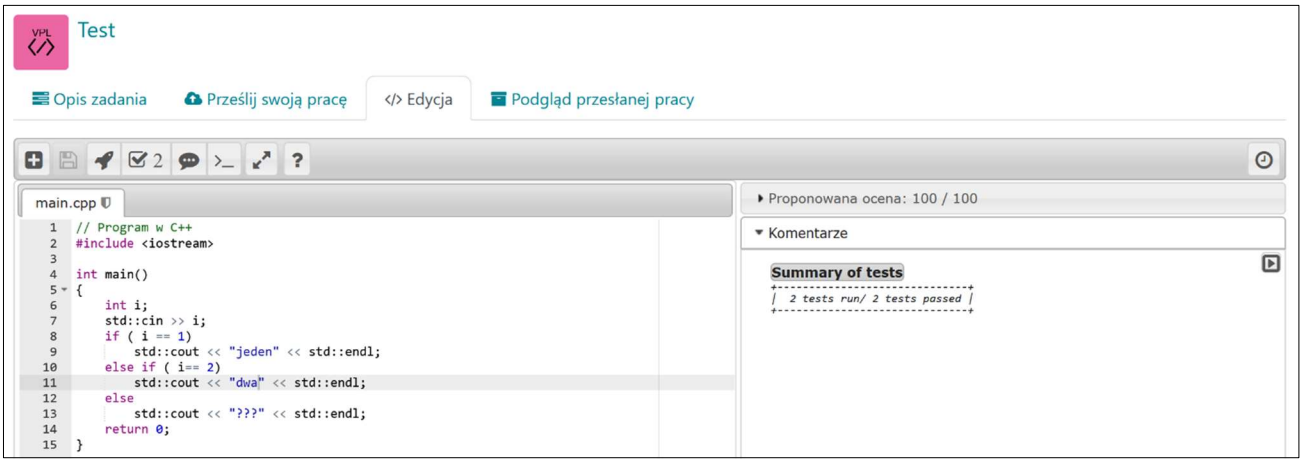


Rysunek 8 Uruchomienie konsoli i wykonanie programu

Po upewnieniu się przez studenta, że wszystko jest dobrze – kompilacja przeszła bez błędów oraz program działa prawidłowo (dla określonych danych wejściowych wyprowadza oczekiwane wyniki) należy przejść do oceny, jak na Rys.9 i Rys.10.

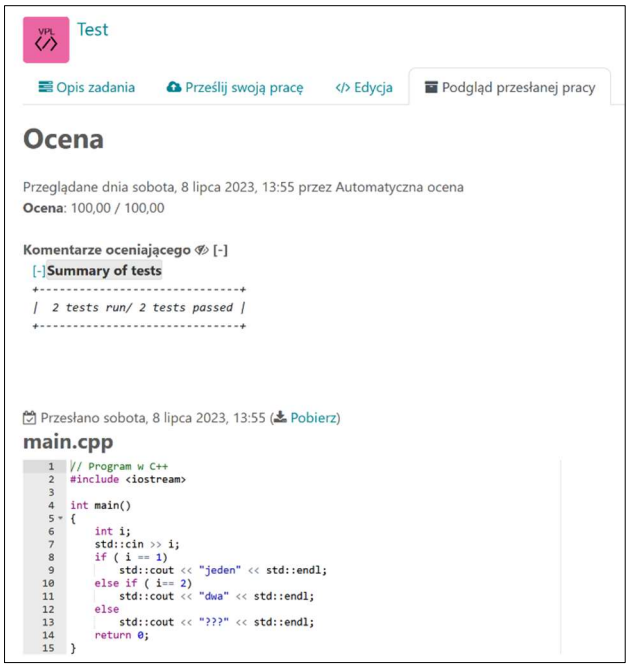


Rysunek 9 Wyniki testów i komentarz do nieudanych prób oraz proponowana ocena



Rysunek 10 Prawidłowe wyniki testów oraz proponowana ocena

Jeśli cały proces przebieg prawidłowo i w zakładce „Podgląd przesłanej pracy” będzie można zobaczyć oceną, komentarz oraz tekst programu, tak jak to pokazano na Rys.11.

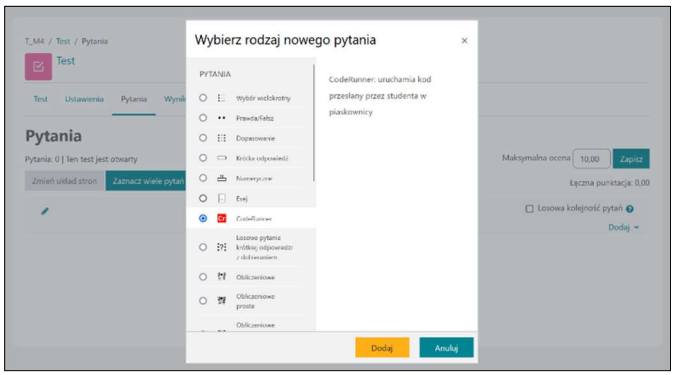


Rysunek 11 Ostateczny wynik pracy nad kodem programu

Wyniki z poszczególnych wirtualnych laboratoriów dla poszczególnych studentów są zapisywane w dzienniku i mogą sumować się z wynikami sprawdzianów wiedzy przeprowadzonych z użyciem aktywności „Testy” oraz możliwe jest też automatyczne wyliczenie oceny na zaliczenie przedmiotu.
Praca nad programem, wszystkie czynności oraz problemy jakie występowały oraz dokładny ich czas są logowane i mogą byś w przyszłości analizowane pod różnym kontem w zależności od potrzeb.
Ocena jest bezstronna i obiektywna, co więcej cały proces bardzo przypomina pracę nad rozwojem oprogramowania w firmach komercyjnych.
Złożoność programów oraz testów i warunków uzyskania punktów może być bardzo złożona i wymagająca, ale powinna być dostosowana za każdym razem indywidualnie do potrzeb danego laboratorium i grupy studentów.

2.2 Pytania typu CodeRunner w testach

Pytania testowe typu CodeRunner są standardowo dostępne w aktywności **Test** Moodle. Służą do sprawdzania wiedzy z zakresy programowania. W trakcie testu obok innych rodzajów testów Rys.12. (wybór wielokrotny, prawda/fałsz, dopasowanie, krótka odpowiedź, numeryczne, esej itd.) można zadać do zrobienia pytanie typu CodeRunner polegające na napisaniu kodu źródłowego realizującego postawiony problem programistyczny i automatycznie je ocenić za pomocą zdefiniowanych testów jednostkowych.



Rysunek 12 Wybór rodzaju pytania w aktywności Test - opcja CodeRunner

Dodając ten typ pytania do testu musimy określić szereg parametrów, takich jak:

- Typ pytania
- Szczegóły pytania zależne od typu pytania
- Ogólne
- Odpowiedzi
- Wstępne ładowanie pola odpowiedzi
- Global extra
- Przypadki testowe w potrzebnej ilości dla danego pytania
- Pliki pomocnicze
- Ustawienia załączników
- Tagi

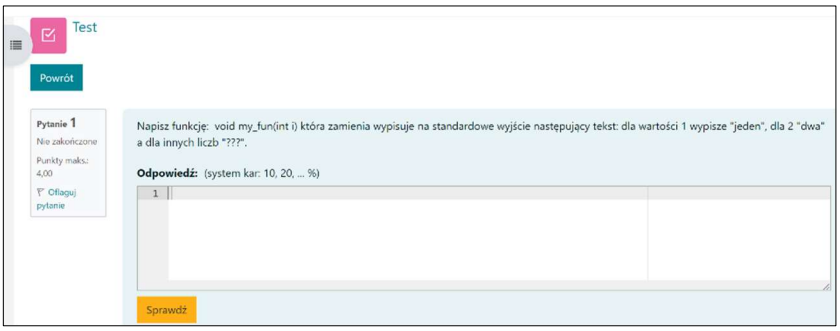
Mogą być następujące typy pytania w pytaniu CodeRunnera:

- Funkcja w języku C
- Program w języku C
- Funkcja w języku C++
- Program w języku C++
- Graf skierowany
- Klasa języka Java
- Metoda języka Java
- Program w języku Java
- Wielojęzykowe
- Nodejs
- Funkcja w języku Octave
- Funkcja w języku Pascal
- Program w języku Pascal
- Php
- Python2
- Python3
- Sql
- Graf nieskierowany

Poszczególne „Przypadki testowe” muszą mieć natomiast określone:

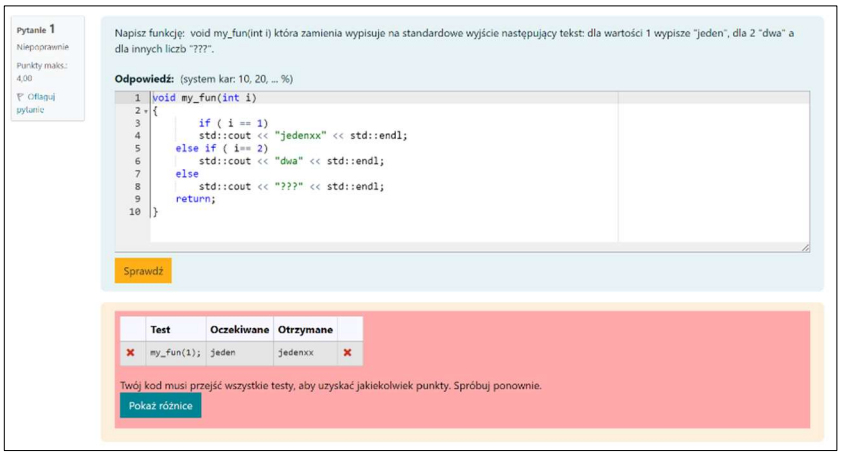
- Standardowe dane wejściowe
- Oczekiwany wynik
- Dodatkowe dane szablonu
- Właściwości testu

Pliki pomocnicze mogą dostarczać danych do testowania nawet w dość dużym rozmiarze.
Oczywiście mysimi też ustawić wszystkie potrzebne opcje dla całego testu. Test może zawierać różnego rodzaju zwykłe pytania oraz wiele pytań typu CodeRunner.
Dokumentacja pakietu CodeRunner znajduje się na GitHub [6].
W trakcie testu w chwili gdy student będzie miał odpowiedzieć na pytanie typu CodeRunner pokaże się na ekranie obraz podobny do przykładowego zaprezentowanego na Rys.13.



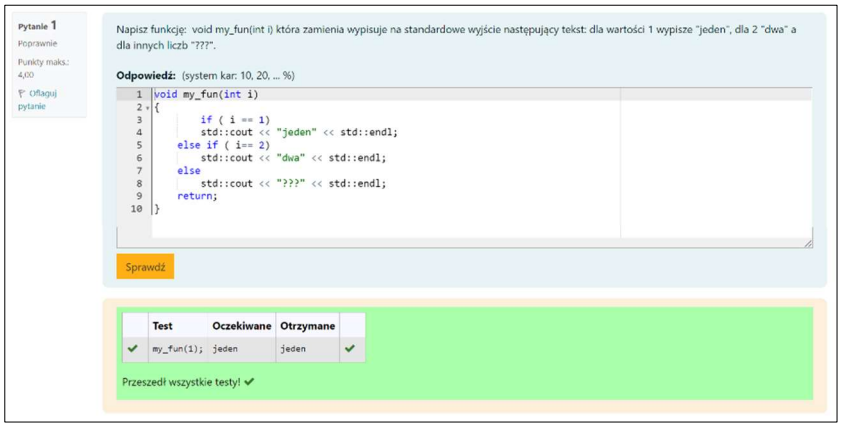
Rysunek 13 Pytanie w CodeRunner

W odpowiedzi powinien napisać tekst funkcji w C++ zgodny z określonym nagłówkiem funkcji oraz realizującą opisane zadanie.



Rysunek 14 Odpowiedź studenta z błędem

Jeśli student po napisaniu kodu źródłowego wybierze przycisk „**Sprawdź**” to jego kod zostanie skompilowany i ewentualne błędy kompilacji zostaną pokazane w oknie poniżej tego przycisku (na czerwonym tle). W takiej sytuacji powinien przeanalizować komunikat błędu i poprawić swój kod źródłowy. Po prawidłowej kompilacji następuje proces sprawdzania działania kodu i jest weryfikowane jego prawidłowe działanie poprzez testy jednostkowe. Jeśli program nie działa prawidłowo to pojawia się okno podobnie jak na Rys.14 i należy przeanalizować problem, dokonać stosownych zmian w kodzie oraz ponownie prawidłowo skompilować. Wtedy ponownie zostaną przeprowadzone testy jednostkowe i w przypadku ich powodzenia pojawi się na ekranie okno podobne jak na Rys.15.



Rysunek 15 Prawidłowa odpowiedź

W teście można umieścić wiele pytań typu CodeRunner, należy jednak pamiętać o uwzględnieniu potrzebnego czasu na ich realizację ponieważ przez swój charakter wymagają one więcej czasu niż inne pytania testowe.

3 Podsumowanie

W artykule przedstawiony dwie możliwości Moodle wspierające proces nauki programowania. Stanowią one bardzo duże udogodnienie dla prowadzącego zajęcia oraz podnoszą jakość kształcenia i jego atrakcyjność dla studentów. Są także namiastką pracy współczesnego programisty nad komercjalnymi rozwiązaniami w komercyjnych firmach rozwijających oprogramowanie w procesie ciągłej integracji (CI) i ciągłego dostarczania (CD) oprogramowania.

Warto również wspomnieć o tym, że w wirtualnym laboratorium programowania istnieje możliwość badania plagiatów w kodzie - zakładka „**Kontrola podobieństw w kodzie**”.

Istotną kwestią jest zgodność wypisywanych przez program komunikatów testowych, które są porównywane z wzorcami. Nawet nieznaczne różnice powodują nie uznanie odpowiedzi, należy zwrócić uwagę studentom, że prace tę wykonuje automat który sprawdza dokładnie całkowitą zgodność i jeśli coś nie jest identyczne nie uznaje tego za prawidłową odpowiedź.

Istnieją również inne możliwości Moodle w tym zakresie, które nie zostały tu omówione, na przykład aktywność **MATLAB Codding Problem**, typ pytań testowych **VPL Question** czy też możliwości aktywności „**Narzędzia zewnętrzne**”. Przy pomocy tej aktywności możemy w ramach Moodle mieć dostęp na przykład do zewnętrznych symulatorów takich jak Marie [7], CircuitVerse [8], Wokwi [9], ARM Thumb (16-bit) Simulator [10] czy innych zdalnych środowisk typu CompilerExplorer [11], OnlineGDB [12], OnlineCompiler [13].

Używając opisanych narzędzi należy zwrócić uwagę na możliwe nadużycia ze strony osób nauczanych. Możliwe jest użycie ChatGPT [14] do pozyskiwania kodu bez jego własnoręcznego napisania. Zaleca się w związku z tym użycia przeglądarki SEB i uniemożliwienie korzystania z innych urządzeń elektronicznych w trakcie zajęć.

4 Bibliografia

[1] vpl.dis.ulpgc.es, dostęp 07.07.2023.

[2] https://moodle.org/plugins/mod_vpl, dostęp 07.07.2023.

[3] https://safeexambrowser.org/news_en.html, dostęp 08.07.2023.

[4] https://vpl.dis.ulpgc.es/documentation/vpl-3.4.3+/, dostęp 08.07.2023.

[5] https://demovpl.dis.ulpgc.es/, dostęp 08.07.2023.

[6] https://github.com/trampgeek/moodle-qtype_coderunner#code-runner, dostęp 08.07.2023.

[7] https://marie.js.org/, dostęp 08.07.2023.

[8] https://circuitverse.org/, dostęp 08.07.2023.

[9] https://wokwi.com/, dostęp 08.07.2023.

[10] https://bkhmsi.github.io/ARMThumb_Sim/#/, dostęp 08.07.2023.

[11] https://godbolt.org/, dostęp 08.07.2023.

[12] https://www.onlinegdb.com/, dostęp 08.07.2023.

[13] https://onecompiler.com/, dostęp 08.07.2023.

[14] https://chat.openai.com/, dostęp 08.07.2023.