Projekt zaliczeniowy na *Podstawy Programowania Seminarium* Kacper Chamera i Aleksander Wyżykowski.

Instrukcja do obsługi programu:
Program działa na podstawie wywołania go i podania odpowiednich argumentów z linii komend. Pierwszym argumentem powinien być odpowiedni świat, np. „qwerty_2". Następnym argumentem powinna być odpowiednia komenda, w zależności co chcemy zrobić w grze. Można podawać dowolną ilość argumentów, rozdzielając je spacją, program wykona je po kolei. Aby zacząć grę należy najpierw utworzyć świat za pomocą komendy *reset* lub *start*. Dostępne są następujące komendy: *start, reset, info, move, explore, right, left* oraz *bot*. Można również skorzystać z komend typu *make* i wykonywać ruch dla świata *„qwerty_2'*:

- *make start* - tworzy mapę o początkowych rozmiarach,
- *make move* - rusza do przodu,
- *make left* - obraca w lewo,
- *make right* - obraca w prawo,
- *make explore* - eksploruje trzy bloki przed czołgiem,
- *make info* - podaje informacje o obecnym położeniu i fazie rozgrywki,
- *make reset* - resetuje świat i tworzy mapę o początkowych rozmiarach,
- *make bot* - uruchamia bota, który automatycznie zwiedza świat,
- *make clean* - czyści wykonywalne pliki, obecnego save'a oraz odpowiedz z API.

Program został podzielony na 5 modułów: *odpowiedź, komunikacja, mapa, bot* oraz *main*. Moduł *komunikacja* jest odpowiedzialny za komunikację z serwerem API i zapisaniem odpowiedzi zwrotnej, a moduł *odpowiedź* korzystając z *komunikacji* wykonuje odpowiednie zapytanie i przetwarza odpowiedź zwrotną do struktury. W module *mapa* znajdują się wszystkie funkcję związane z dalszym użytkowaniem programu, a więc: alokowanie, tworzenie, zapisanie, wypisywanie i wczytywanie świata oraz funkcje potrzebne do aktualizowania rozmiarów i elementów odkrytego świata, jak i zwalniania niepotrzebnych struktur. W module *bot* został zawarty algorytm pozwalający na automatyczne odkrywanie świata. Korzysta on z funkcji napisanych już w innych modułach. Moduł *main* odpowiada natomiast za wywoływanie wszystkich funkcji w prawidłowej kolejności i reguluje całe działanie programu.

Wszystkie takie moduły zawierają swoje biblioteki z funkcjami, które są naprzemiennie zawierane w innych modułach. Program został podzielony na moduły tak aby uzyskać jak najwyższą przejrzystość kodu i żeby całość sprawiała

wrażenie zwartej i logicznej całości. W programie zostały użyte dwie struktury, które zawierają wszystkie możliwe informacje o światach na serwerze oraz lokalnym zatem: obecne położenie, kierunek, ilość kroków, obecną sesję, itp. Pierwsza z nich - *struct mapa* odpowiada za przechowywanie informacji zwróconych z jSONA. Druga z nich - *struct mapa_dynamiczna* przechowuje informacje o mapie lokalnej oraz tego czego nie ma w pierwszej strukturze, czyli delty położenia oraz rozmiarów mapy. Gra komunikuje się z serwerem typu API, do którego wykonuje zapytanie o zwrot odpowiednich informacji, które potem uzyskuje i zapisuje je aż do kolejnego zapytania do serwera.

Scenariusz działania programu:

1. Utworzenie odpowiednich struktur do przechowywania informacji.
2. Wejście w pętle *for* w celu wykonania odpowiedniej liczby komend przy wywołaniu programu.
3. Wykonanie odpowiedniego zapytania w przypadku komendy:
    1. info: zapytanie do serwera, odczyt informacji, wypisanie ich na ekranie;
    2. start oraz reset: wykonanie odpowiedniego zapytania, zapis informacji zwrotnej, wyświetlenie oraz zapisanie lub nadpisywanie nowej mapy.
    3. move: odpowiednie zapytanie do serwera, przypisanie informacji zwrotnych, aktualizacja obecnej mapy, sprawdzenie czy czołg znajduje się na jej krańcu; jeżeli tak - rozszerzamy ja w odpowiednim kierunku, jeżeli nie - zostawiamy „starą” strukturę po czym niezależnie wyświetlamy i zapisujemy najnowszą mapę.
    4. explore: działa jak ww. komenda z wyjątkiem sprawdzania warunku krańca mapy.
    5. right oraz left: wykonują odpowiednie zapytanie, po czym przekręca czołg w odpowiednim kierunku, informując nas o tym na ekranie.
    6. bot: pobiera informacje o obecnym położeniu, rusza do przodu w poszukiwaniu ściany, a gdy ją znajdzie rozpoczyna algorytm odkrywający kontury mapy; w przypadku dojścia do położenia początkowego algorytm się kończy.
4. Na samym końcu zwalniamy pomocnicze struktury oraz inne zmienne i zwracamy wartość końcową funkcji *main*.