Motorola Science Cup²
Zadania II Etapu

Interpreter języka ATARI Logo

OMÓWIENIE

Celem tego zadania jest stworzenie własnego interpretera języka ATARI Logo wypuszczonym na rynek w 1983 r. Język ten może być szczególnie przydatny do nauki programowania dla najmłodszych. W implementacji ATARI na środku ekranu pojawia się przyjazny żółw, którym młody programista może dowolnie sterować wpisując odpowiednie komendy. Żółw ten porusza się po ekranie rysując za sobą linię.

Przydatne materiały:

- Książka Prof. Ryszarda Tadeusiewicza „Atari Logo Komputerowe Przygody” (dostępna na jego stronie: https://www.cri.agh.edu.pl/uczelnia/tad/dorobek_naukowy.php?id=pubwww)
- Przykładowy internetowy logo interpreter: https://www.calormen.com/jslogo/#

W zadaniu tym prosimy o dostarczenie interpretera z interfejsem graficznym. Będzie to:

- plansza o zadanych wymiarach na której żółw (lub inny obiekt) będzie dla nas rysował
- miejsce gdzie można wpisywać kod, który będzie interpretowany na planszy
- miejsce, gdzie będzie wyświetlany ewentualny output tekstowy, błędy - konsola
- wszystkie inne zaprojektowane przez Was przyciski

Aplikacja może być zrealizowana w dowolnym języku programowania, ale musi uruchomić się na systemie Windows 10. Aplikacja powinna zapisywać dane lokalnie.

Jako rozwiązanie zadania prosimy o przesłanie:

- kodu źródłowego (wraz z testami automatycznymi)
- dokładnej instrukcji opisującą proces budowania gotowej do uruchomienia aplikacji

- gotowej aplikacji
- pozostałej dokumentacji

Prace, które będą dostarczone bez kodu źródłowego i nie uruchomią się przy użyciu dostarczonego installera (i/lub pliku wykonywalnego) nie będą oceniane. Aplikacja będzie uruchamiana na komputerze pozbawionym dostępu do internetu, na Windowsie 10 bez zainstalowanych narzędzi deweloperskich, dostępna przeglądarka: Chrome. Jeśli do uruchomienia interpretera będą potrzebne jakieś dodatkowe narzędzia lub konfiguracja środowiska, aplikacja w ramach instalacji powinna to zapewnić.

CO OCENIAMY

Oceniać będziemy:

1. Zgodność rozwiązania ze specyfikacją (zobacz opisy poszczególnych etapów zadania)
2. Zaimplementowaną architekturę aplikacji, czystość kodu
3. Dokumentację projektu
4. Zaproponowany interfejs graficzny aplikacji
5. Testy automatyczne

ETAPY ZADANIA

Etap I - Wersja podstawowa

Za ten etap zadania uzyskacie do **40** punktów.

Opis:

- Stwórzcie interpreter z interfejsem graficznym, żółwiem lub dowolnym innym obiektem na środku planszy pokrywającej cały ekran (min 800x600px), gdzie w osobnym obszarze można wpisywać podane komendy:
    - CS (clear screen) - usunięcie rysunków / czyszczenie planszy
    - HT (hide turtle) - ukrywanie ikony żółwia
    - ST (show turtle) - pokazywanie ikony żółwia
    - RT kąt (right turtle) - obrót żółwia o zadany kąt (podany w stopniach) w prawo
    - LT kąt (left turtle) - obrót żółwia o zadany kąt (podany w stopniach) w lewo
    - FD kroki (forward) - ruch żółwia o zadaną ilość kroków do przodu
    - PU (pen up) - poruszający się żółw nie zostawia śladu

    - PD (pen down) - poruszający się żółw zostawia ślad
    - BK kroki (backwards) - cofnięcie się o wskazaną liczbę kroków
- Komendy można dowolnie łączyć np:

  Żeby narysować kwadrat w odpowiednie pole interpretera należy wpisać:

  CS FD 50 LT 90 FD 50 LT 90 FD 50 LT 90 FT 50 LT 90
- Zamplementujcie komendę REPEAT
    - REPEAT x [abc] - powtórzyć x razy instrukcje abc w nawiasach kwadratowych
    - np. REPEAT 4 [FD 50 LT 90] narysuje kwadrat
- Pamiętajcie o walidacji składni - odpowiedni komunikat na konsolę, gdy program zawiera błąd
- Dodajcie przycisk HELP, pod którym opisane są zaimplementowane przez Was komendy

Etap II - Wersja rozszerzona - punkty dostępne po zrealizowaniu założeń poziomu podstawowego

Ten etap wart jest do **20** punktów.

Opis:

- Dopiszcie możliwość tworzenia procedur:

  ```
  TO NAZWA_PROCEDURY
          .....
  END
  ```

  Żeby można było napisać np:

  ```
  TO OKRAG
          REPEAT 36 [FD 6 RT 10]
  END
  ```
- Zaimplementujcie **ERALL** - usuwanie procedur
- Dodajcie możliwość dodawania parametrów np.

  ```
  TO TROJKAT :BOK
          REPEAT 3 [FD :BOK LT 120]
  END
  ```
- Pamiętajcie o obsłudze, gdy użytkownik poda złą ilość parametrów
- Zaimplementujcie możliwość pojawienia się większej ilości żółwi i sterowanie nimi komenda **TELL,** przykłady:

    - TELL 1 - pojawienie się nowego żółwia (pierwszy ma indeks 0), wszystkie komendy pojawiające się po komendzie TELL dotyczą wezwanego żółwia, aż do kolejnego wywołania TELL
    - TELL [0 2] - wskazywanie kilku żółwi jednocześnie
- Do wykonywania konkretnych rozkazów dla konkretnych żółwi zaimplementujcie komendę **ASK numery rozkazy**, przykłady:

  > **ASK** [1] [LT90]

Etap III - Operacje dodatkowe - punkty dostępne po zrealizowaniu założeń poziomu rozszerzonego

Ten etap wart jest do **40** punktów.

Opis:

Użytkownikom waszego interpretera bardzo podobają się dotychczas wykonane funkcje. Użytkownicy proszą o dodanie kilku dodatkowych funkcjonalności:

- Żółwie mogą zmieniać swoje kolory. Do zmiany koloru żółwia zaimplementujcie komendę **SETC kolor**, przykłady:
    - SETC 6 - dostępnych jest 128 kolorów

  Definicje kolorów znajdziecie w książce Atari Logo Komputerowe Przygody
- Każdy żółw może używać jednego z trzech pisaków (0, 1, 2). Do wybierania pisaka którym posługują się żółwie zaimplementujcie komendę **SETPN pisak**, przykłady:
    - SETPN 2
- Każdy pisak może mieć zdefiniowany kolor. Do definicji kolorów pisaka zaimplementujcie komendę **SETPC pisak kolor**, przykłady:
    - SETPC 1 16 - dla pisaka 1 ustaw kolor 16 (dostępnych jest 128 kolorów)
- Każdy żółw ma swój identyfikator (indeks). Zaimplementujcie komendę **WHO** do pobierania identyfikatora żółwia. Komenda WHO będzie mogła być użyta do budowania złożonych wyrażeń.
- Po zastosowaniu komendy TELL stworzymy grupę żółwi, które będą wykonywały kolejne rozkazy. Zaimplementujcie komendę **EACH [komenda]** aby komenda była wykonywana sekwencyjnie jedna po drugiej dla grupy żółwi. Przykłady:
    - EACH [REPEAT 4 FD 10*WHO RT 90]

      Zwróćcie uwagę że wyrażenie 10*WHO wymaga uwzględnienia interpretacji wyrażeń arytmetycznych.

- Do edycji procedur służy komenda ED. Komenda ED przestawia interpreter w tryb edycji procedury. W trybie edycji procedur interpreter staje się edytorem procedur. Zapis zdefiniowanej procedury dokonuje się automatycznie po wyjściu z trybu edycji.
  Zaimplementujcie komendę **ED listanazw**, przykłady:
  - ED KWADRAT (edycja procedury KWADRAT)
  - ED [KWADRAT TROJKAT] - (edycja procedur KWADRAT i TROJKAT)

  Wyjście z trybu edycji możecie zaimplementować w wybrany przez siebie sposób.
- Zaimplementujcie komendę **POTS** która pokaże nazwy wszystkich procedur wraz z ich parametrami.
  Wynikową listę możecie wyświetlić w wybrany przez siebie sposób.
- Po zamknięciu programu wszystkie procedury zostaną usunięte. Dopiszcie możliwość wykonywania zapisu swojego programu w celu umożliwienia pracy nad nim w przyszłości. Użytkownicy chcieliby mieć możliwość zapisywania do pliku oraz wczytywania z pliku zdefiniowanych procedur.
  Do wczytania procedur z pliku zaimplementujcie komendę **LOAD plik**, przykłady:
  - LOAD c:\bear.logo

  Do zapisu procedur do pliku zaimplementujcie komendę **SAVE plik**, przykłady:
  - SAVE c:\bear.logo
- Dopiszcie możliwość wykonywania zapisu obrazu planszy do pliku w dowolnym momencie działania aplikacji.

Zwróćcie uwagę, że w niektórych miejscach nie sprecyzowano czy dana funkcjonalność ma być wywołana komendą czy przez jakiś element interfejsu użytkownika. Potraktujcie wszystkie takie miejsca jako możliwość dowolnej realizacji. Podobnie potraktujcie kwestię wyświetlania komunikatów. Wierzymy, że znajdziecie dobre rozwiązania dla takich elementów.

Wykorzystane źródła:

1. Ryszard Tadeusiewicz ,,Atari Logo Komputerowe Przygody” https://www.cri.agh.edu.pl/uczelnia/tad/dorobek_naukowy.php?id=pubwww