# Projekt na seminarium z podstaw programowania

*SIO - Symulator Inteligentnego Odkurzacza*

Wstęp

Program podzielony jest na moduły: main, mapa, polecenia i logika. Każdy z nich ma swoje zastosowanie.

Moduł mapa tworzy obszar, w którym będzie poruszał się później nasz odkurzacz, zapisuje ją też do pliku, przechowuje ją i alokuje pamięć.

Moduł polecenia ma za zadanie komunikować się z serwerem i zwracać odpowiedź o stanie pól, przetwarza odpowiedź jaką dostaje na język programowania czytelny dla programu i za pomocą biblioteki cJSON zwraca wynik jako dane.

Kolejny moduł - logika, interpretuje położenie odkurzacza, porusza się nim po mapie za pomocą algorytmu.

Wszystkie te moduły podpięte są pod plik main.c, który zawiera również biblioteki używane w programie razem z biblioteką cJSON.

Program zawiera również plik makefile, który ułatwia uruchomienie go.

Cel i działanie programu

Program został stworzony w celu automatycznego odkrycia mapy gry "Logo Worlds" dostępnej na stronie <http://edi.iem.pw.edu.pl:30000/index.html>.

Aby uruchomić program, najpierw trzeba sprawdzić, czy zainstalowana biblioteka cJSON jest poprawnie zaadresowana. Kiedy wszystko zostało sprawdzone, możemy skompilować program używając "make".

Skompilowany program możemy używać manualnie lub sprawić, żeby działał automatycznie.

Jeśli chcemy używać go manualnie, w konsoli musimy wpisać "./projekt" i po spacji wypisywać komendy, które program ma wykonać. Dostępne komendy:

M (ruch) - sprawia, że czołg rusza się do przodu,

L (lewo) - obraca czołg w lewo,

R (prawo) - obraca czołg w prawo,

E (odkryj) - odkrywa 3 pola w prostopadłej linii do kierunku czołgu.

Struktury

W module mapa.h program zawiera wszystkie używane struktury.

Dane, do której są zapisywane informacje jSON, po poleceniach ruch, lewo, prawo lub info. Zawierają informacje takie jak aktualna pozycja, rodzaj pola na którym jest czołg i kierunek, w który jest zwrócony.

Dane2, do tej struktury zapisywane są dane jSON po funkcji explore, są to współrzędne trzech pól oraz ich rodzaj.

Pole – struktura, do której przekazujemy wszystkie najważniejsze elementy, opisujące postęp odkrycia mapy.

* char \*\*index – tablica dwuwymiarowa, w której zapisujemy znaki, które pozwalają nam wizualizować mapę.

Legenda znaków:

. - nieodkryte pole,

w - ściana (wall),

g – trawa (grass),

s – piasek (sand),

G – pole z trawą zajęte przez czołg,

S – pole z piaskiem zajęte przez czołg.

Struktura pomocnicza Rozmiar zapisuje wartości X i Y, które określają wielkość końcową mapy po której porusza się nasz czołg.

Algorytm

Aby uruchomić algorytm, który sam będzie odkrywał mapę należy po skompilowaniu programu wpisać "*./projekt SIOv2.0*" / “./projekt SIOv3.0” lub "*make test-SIOv2.0*" / “*make test-SIOv3.0*”.

Algorytm posiada 2 stopnie zaawansowania:

* 1 stopień (SIOv2.0) odkrywa mapę do pewnego momentu i wtedy się przerywa,
* 2 stopień (SIOv3.0) odkrywa mapę do momentu, gdy cała mapa nie zostanie odkryta.

Algorytm sprawi, że czołg zacznie się poruszać w poszukiwaniu pierwszej ściany. Kiedy na nią trafi, będzie się poruszał wzdłuż ściany aż nie wróci na pozycję wyjściową. Po wykonaniu tej czynności obróci się o 180° i sprawdzi czy cała mapa została odkryta. Jeśli mapa została odkryta algorytm się skończy i mapa zostanie zapisana do pliku tekstowego. Jeśli zostały nieodkryte pola algorytm będzie się powtarzał do momentu kiedy wszystkie pola zostaną odkryte.

Algorytm jest złożony w większości z funkcji rekurencyjnych, które odpowiadają za ruch czołgu.

**do\_przodu**

Czołg rusza się do przodu:

* Jeśli trafi na ścianę, funkcja się przerywa i zaczyna się dalszy etap algorytmu,
* Jeśli się ruszy, funkcja jest wywoływana rekurencyjnie do momentu aż czołg trafi na ścianę.

Zwraca strukturę Pole.

**lewo\_ruch**

Jedna z dwóch składowych funkcji rekurencyjnych, które zostały napisane w celu wprawienia czołgu w ruch wzdłuż ściany w lewą stronę. Na samym początku przekazujemy zapisane wcześniej koordynaty pola, na którym czołg zaczął wykonywać algorytm obchodzący ścianę.

Funkcja wykonuje 3 polecenia:

* lewo,
* ruch,
* odkryj.

Po tym sprawdza czy czołg ruszył się z miejsca:

* Jeśli się ruszył wykonuje się funkcja prawo\_ruch
* Jeśli został w miejscu ponownie wykonuje się lewo\_ruch

Po skończeniu rekurencji zwraca strukturę Pole.

**prawo\_ruch**

Druga funkcja rekurencyjna razem z lewo\_ruch. Funkcja kończąca pętlę rekurencyjną.

Wykonuje 3 polecenia:

* prawo,
* ruch,
* odkryj.

Po wykonanych zadaniach sprawdza, czy czołg wrócił na początkową pozycję. Jeśli wrócił funkcja zwraca strukturę Pole i po kolei wszystkie funkcje zaczynają zwracać strukturę do samego początku.

Jeśli pierwszy warunek nie został spełniony rekurencja trwa dalej.

* Jeśli czołg się ruszył znowu wykonuje się prawo\_ruch,
* Jeśli został w miejscu funkcja wywołuje lewo\_ruch.

**znajdz\_mur\_zewnetrzny**

Funkcja, sprawdza czy na bieżącym etapie odkrywania mapy został odkryty cały mur okalający obszar, w którym porusza się czołg.

Jeśli znajdzie taki mur, zostaną zwrócone wymiary mapy do struktury pomocniczej Rozmiar.

**sprawdz\_postep**

Funkcja pomocnicza, sprawdza czy cała mapa została odkryta. Jest zainicjalizowana jako warunek dla pętli while, która powtarza algorytm do momentu odkrycia całej mapy. Zwraca int.

**obrot\_180**

Wykonuje 2 razy polecenie prawo.

Zwraca strukturę Pole.

Autorzy:   
Aleksandra Hermańska  
Marcin Pietrzak