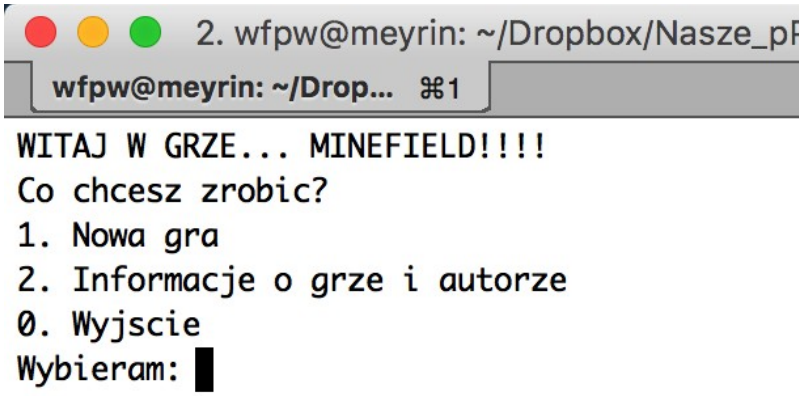
Podstawy programowania, Poniedziałek 30.05.2016, 8-10
Projekt, część 1

1. Zadanie
Projekt polega na stworzeniu logicznej gry komputerowej działającej w trybie tekstowym o nazwie „Minefield”.

2. Cele
Celem projektu jest napisanie bardziej rozbudowanego programu niż było to możliwe na pojedynczych zajęciach i przećwiczenie wcześniej nauczonych mechanizmów języka C. Ponieważ jest to projekt, wymagana jest również własna kreatywność oraz inwencja – nie wszystko w treści zadania będzie napisane krok po kroku jak to zrobić.
Uwaga! Zalecane jest szerokie stosowanie zewnętrznych funkcji – w dobrym projekcie funkcja `main` powinna być jak najkrótsza.

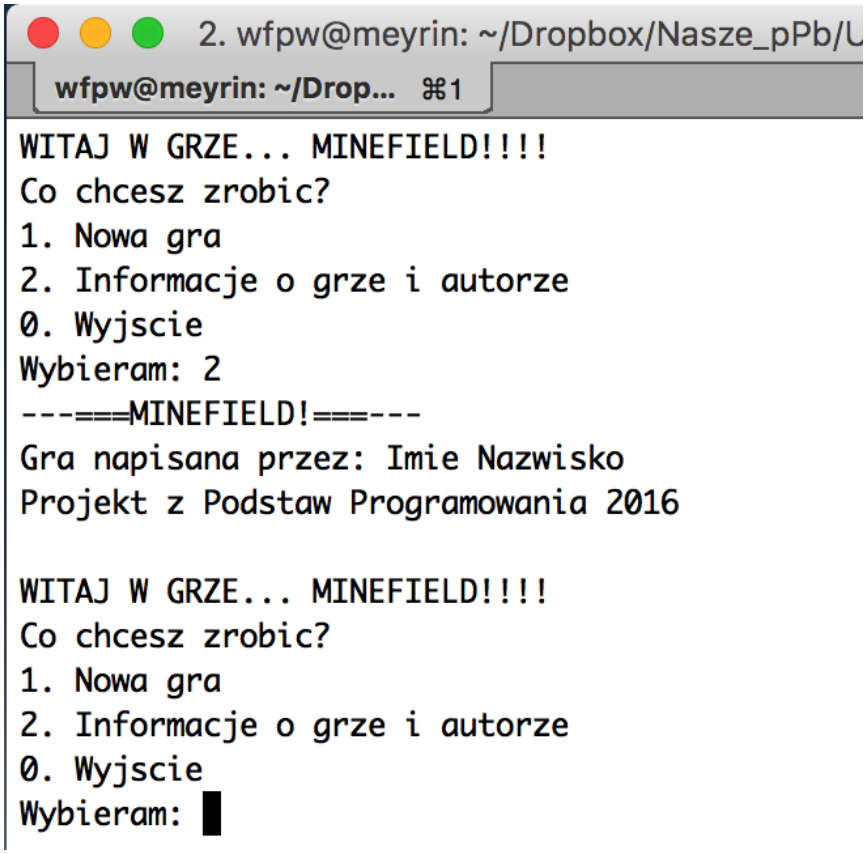
3. Menu główne
Od razu po uruchomieniu program powinien wyświetlić menu główne, w którym znajdować się będzie opis możliwych do wyboru opcji. Menu główne powinno wyglądać mniej więcej tak:



Menu działa w nieskończonej pętli, której przerwanie i wyjście z programu następuje po wpisaniu opcji „0”. Wybór każdej z opcji jest dokonywany poprzez instrukcję `switch-case`.

2. Informacja o autorze

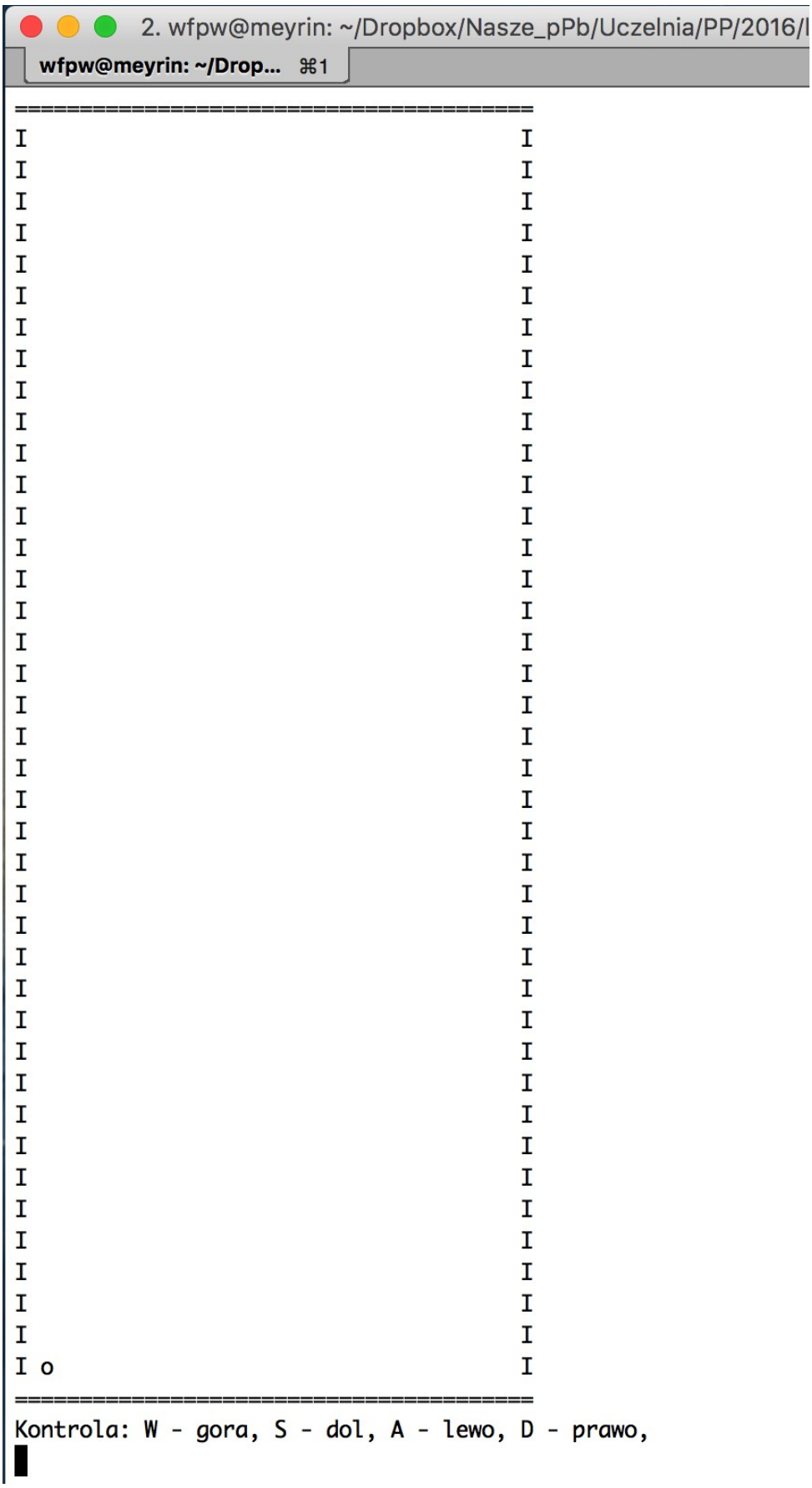
Wybranie opcji „2” powinno wyświetlić na ekranie informację o autorze a następnie wrócić do menu głównego:



3. Plansza

Główną częścią pierwszych zajęć projektowych jest stworzenie planszy, w której poruszał się będzie saper. Saper powinien mieć możliwość poruszania się przy pomocy klawiszy „W”, „S”, „A”, oraz „D”. Powinien on być oznaczony na planszy za pomocą literki „o”.

Przykładowa plansza przedstawiona jest na poniższym rysunku:


2. wfpw@meyrin: ~/Dropbox/Nasze_pPb/Uczelnia/PP/2016/l
wfpw@meyrin: ~/Drop... ⌘1
Kontrola: W - gora, S - dol, A - lewo, D - prawo,

Szczegóły implementacji
Głównym zadaniem jest implementacja funkcji `void NowaGra()`, która jest wywoływana po wybraniu w menu opcji 1. W funkcji tej tworzymy planszę planszę oraz dodajemy ruch sapera.

1) Stworzenie sapera:
Tworzymy strukturę `Punkt`, która zawiera dwa pola (jak łatwo się domyślić, przechowujące dwuwymiarowe współrzędne położenia sapera):
```
int x;
int y;
```
Do jej obsługi wystarczy jedna fukcja:
```
void UstawPunkt(Punkt *p, int X, int Y)
```
Struktura ma swoje pliki `.h` i `.c`.
W funkcji `void NowaGra()` tworzymy obiekty typu `Punkt` ustawiając mu pozycję startową sapera (odpowiednie współrzędne odpowiadające lewemu dolnemu rogowi planszy)

2) Narysowanie planszy:
Rozwijamy funkcję `void NowaGra()`. Planszę można zaimplementować następująco:
- zakładamy stały rozmiar planszy (w naszym przypadku 40) – ponieważ tę wartość będziemy wykorzystywać często, można po prostu zdefiniować stałą globalną, czyli od razu pod „include'ami” wstawiamy:
```
#define PLANSZA 40
```
- tak zdefiniowaną stałą możemy używać już dowolnie w całym pliku, np.:
`for(int i=0;i<PLANSZA;i++)` - wykona 40 iteracji
- „rysujemy” planszę - przedstawiona powyżej plansza jest wynikiem zastosowania odpowiedniej kombinacji funkcji `printf` umieszczonych wewnątrz dwóch zagnieżdżonych pętli `for`. Dla każdego punktu (x,y) planszy (każdej iteracji) musimy sprawdzić, czy odpowiada on współrzędnym położenia sapera. Jeśli dany punkt planszy nie odpowiada położeniu sapera, wypisujemy na ekran spację, jeśli zaś odpowiada, to wypisujemy „o” w danym punkcie → najprościej wszystko to zaimplementować w zewnętrznej funkcję `void plansza(Punkt *s)`, która rysuje ramkę z pozycją sapera. Funkcję tę wywołujemy wewnątrz funkcji `void NowaGra()`

3) Ruch sapera:
Ruch sapera polega na zmianie położeń (czyli wartości x oraz y) sapera `s` (obiektu typu `Punkt`) oraz przerysowywaniu planszy po każdej takiej zmianie. W tym celu należy w funkcji `void NowaGra()` zaimplementować nieskończoną pętlę `while`, w której każda zmiana pozycji będzie powodować czyszczenie ekranu terminala (komenda `system("clear")` – patrz Uwaga poniżej i zajęcia 10) oraz ponowne narysowanie planszy poprzez wywołanie funkcji `void plansza(Punkt *s)`. Jedyną niewiadomą pozostaje zmiana pozycji sapera – jak to zrobić? Do tego celu należy wykorzystać funkcję `char getch()` z pliku `getchar.h` (jest to podobna funkcja do `scanf`, z tą różnicą, że nie wymaga potwierdzenia wprowadzenia znaku poprzez wciśnięcie klawisza Enter).
Przykład wykorzystania funkcji `char getch()` - wczytanie pojedynczego znaku z klawiatury i jego wypisanie na ekran:
```
#include <stdio.h>
#include "getchar.h"

int main(void)
{
  char znak;
  znak = getch();
  printf("Podany znak to: %c\n",znak);
```

```
  return 1;
}
```
W przypadku ruchu sapera, wciśnięcie klawiszy:
„w” - przesuwa sapera o 1 w górę (czyli wartość y położenia sapera zmniejsza się o 1)
„s” - przesuwa sapera o 1 w dół (czyli wartość y położenia sapera zwiększa się o 1)
„a” - przesuwa sapera o 1 w lewo (czyli wartość x położenia sapera zmniejsza się o 1)
„d” - przesuwa sapera o 1 w prawo (czyli wartość x położenia sapera zwiększa się o 1)
„0” - wychodzi z gry (przerywa pętlę przerysowującą planszę) i wraca do menu głównego

Uwaga!
Bardzo użyteczna komenda:
`system("clear");`
z biblioteki **stdlib.h**
na systemach UNIX'owych czyści całe okno terminala: umożliwia to np. zastępowanie jednego menu innym.

Punktacja:

1. Stworzenie menu głównego (**1 p.**)
2. Obsłużenie opcji wyjścia i informacji o autorze (**1 p.**)
3. Obsłużenie opcji „Nowa gra” wraz ze stworzeniem struktury `Punkt` i narysowaniem planszy (**6 p.**)

Całość kompilujemy w linii poleceń używając pliku `Makefile` oraz polecenia `make`. (**1 p.**)

Zrozumienie, analiza treści, projekt programu + poprawność i estetyka kodu. (**1 p.**)

Dodatkowo – tworzymy „ścieżkę bezpieczeństwa” - czyli oznaczamy pola w planszy, które odwiedził saper. Jak to zrobić?

- musimy zapamiętywać wszystkie poprzednie ustawienia sapera w kolejnym ruchu
- tworzymy globalną (od razu pod „include’ami”) tablicę 2D – planszę naszej gry, np. w następujący sposób (zakładamy rozmiar 40x40):

```
#define PLANSZA 40
bool tablica[PLANSZA][PLANSZA];
```

- w funkcji `void NowaGra()` (przed albo po stworzeniu sapera) ustawiamy wszystkie jej pola na `false`
- dla pozycji sapera zmieniamy w tablicy dla danego położenia `false` na `true`, czyli jeśli będziemy się przesuwać po planszy, to kolejne pola będą zmieniane na `true`. W każdym przerysowaniu planszy sprawdzamy tablicę i jeżeli dla danego (x,y) jest tam `true`, zamiast spacji wstawiamy znak `#`