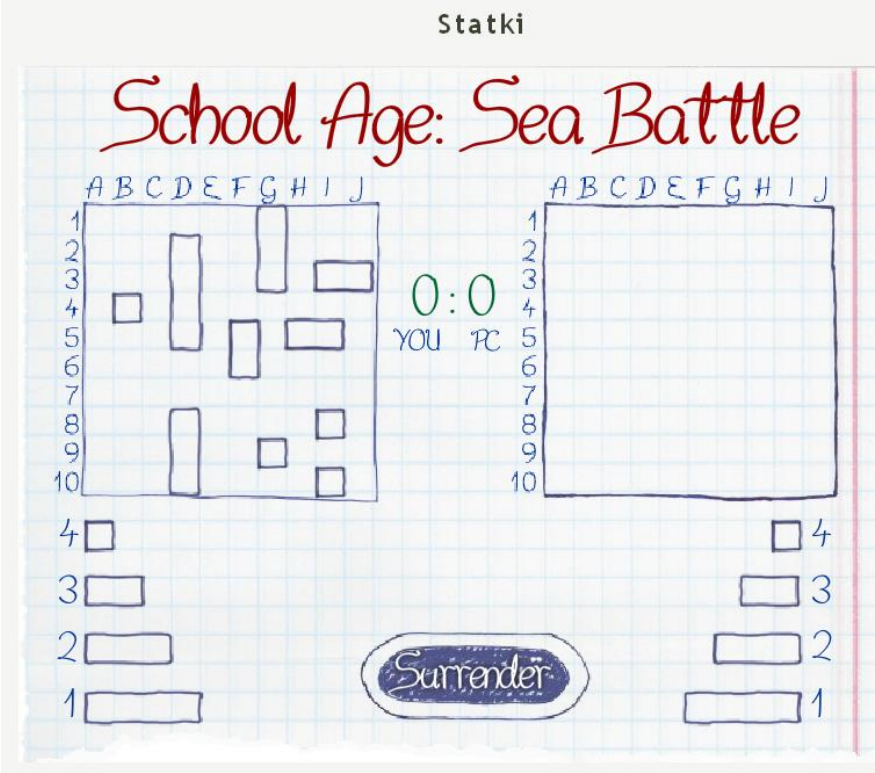
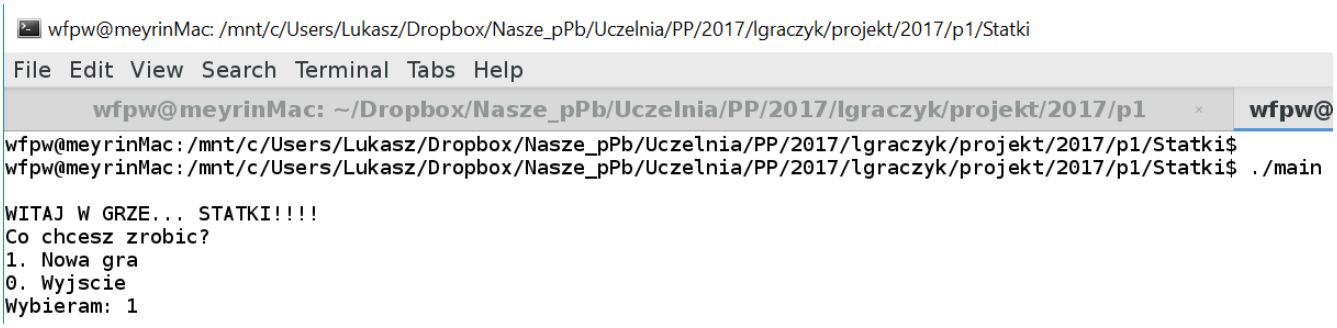
Podstawy programowania, Poniedziałek 4.06.2018
Projekt (zajęcia kontrolne 3), część 1

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



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

2. Nowa Gra

Główną częścią pierwszych zajęć projektowych jest stworzenie funkcji `void NowaGra()`, w której ustawimy parametry jednego gracza (każdego z Was) – czyli statki na planszy.

Szczegóły implementacji

Głównym zadaniem jest implementacja funkcji `void NowaGra()`, która jest wywoływana po wybraniu w menu opcji 1. W funkcji tej tworzymy dwie plansze (naszych statków oraz trafień w plansze przeciwnika) oraz dodajemy statki do naszej planszy.

Struktura `Gracz`***:***

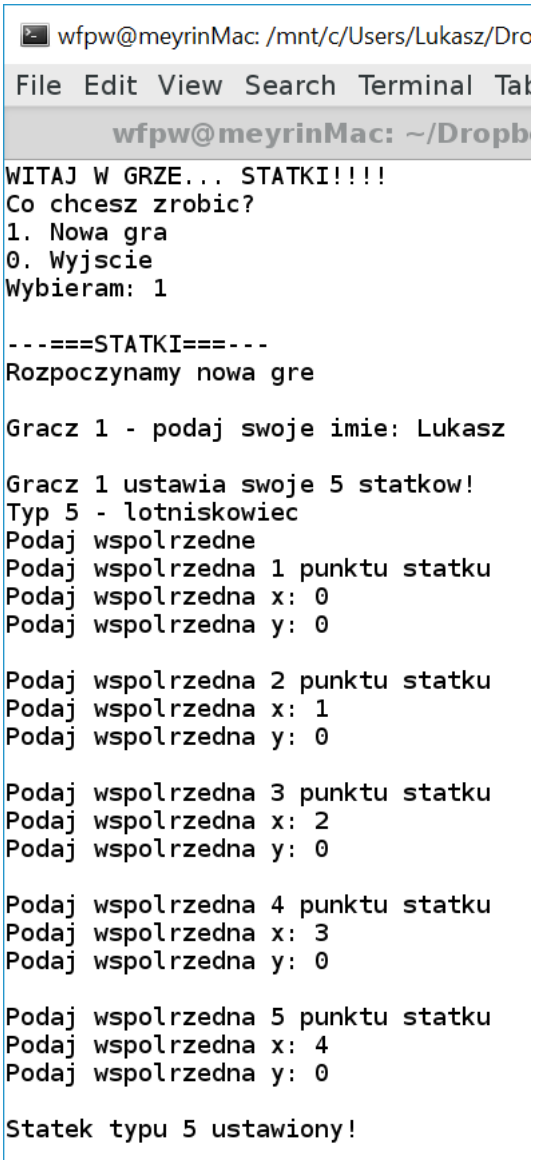
Tworzymy strukturę `Gracz`, która zawiera trzy pola: imię gracza (tablica znaków), oraz dwie tablice dwuwymiarowe (statyczne) typu `int` o rozmiarze 10x10 każda (jedna do przechowywania planszy z naszymi statkami, druga do przechowywania planszy z trafieniami w statki przecownika).

Do „obsługi” gracza tworzymy trzy funkcje pomocnicze:

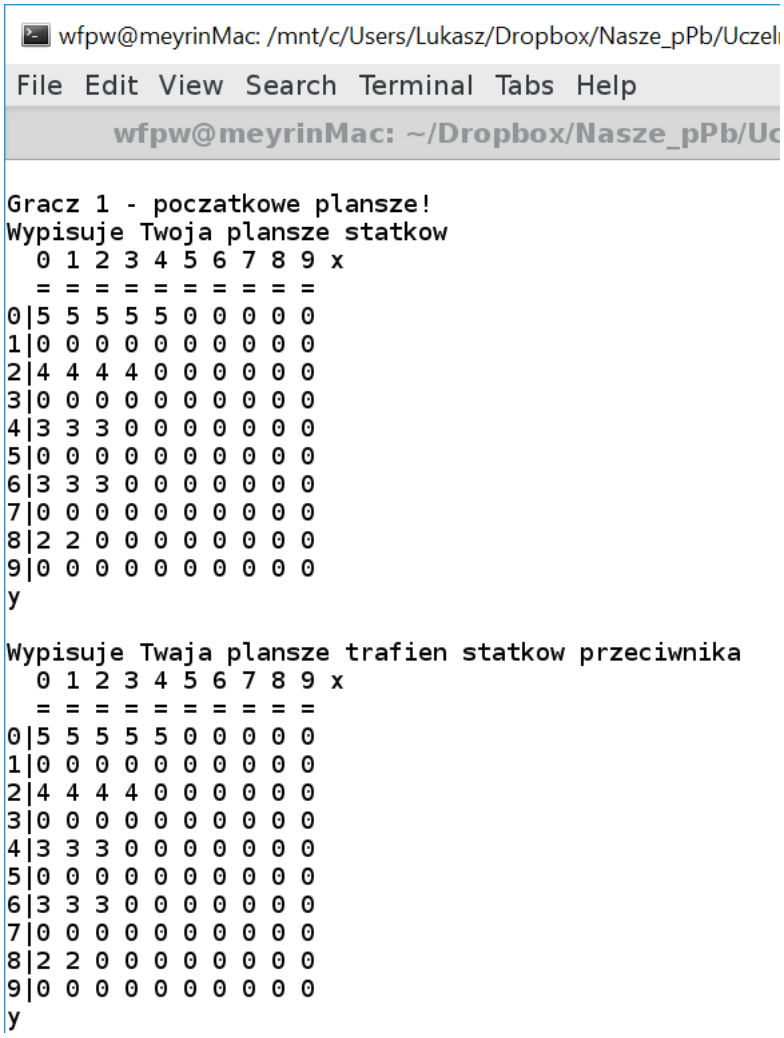
- `void InicjalizujGracza(Gracz *gracz)` – przyjmuje zmienną typu `Gracz` i pyta użytkownika o podanie z klawiatury imienia gracza oraz ustawia wszystkie wartości pól obu tablic na 0

- `void UstawStatek(Gracz *gracz, int typ)` – przyjmuje zmienną typu `Gracz` i ustawia na naszej własnej planszy statek danego typu. Typ odpowiada danemu statkowi oraz jego rozmiarowi:
  - Typ 5 – lotniskowiec – 5 kratek
  - Typ 4 – pancernik – 4 kratki
  - Typ 3 – krążownik – 3 kratki
  - Typ 2 – niszczyciel – 2 kratki

Funkcja pyta się użytkownika o podanie z klawiatury pozycji (x,y) każdej kratki statku i ustawia odpowiednie pola na wartość odpowiadającą danemu typowi (np. dla lotniskowca ustawiamy 5 kratek na wartość 5). Przykładowo:



- `void WypiszPlansza(Gracz *gracz, int typPlanszy)` – przyjmuje również zmienną typu `Gracz` oraz zmienną typu int określającą typ planszy, którą chcemy wypisać (0 – własnych statków, 1 – planszę trafień). Przykład użycia:



Funkcje te wykrozystujemy do inicjalizacji statków gracza oraz wypisania jego plansz (np. w funkcji `NowaGra`)

W grze ustawiamy każdemu graczowi 5 statków:

- 1 lotniskowiec
- 1 pancernik
- 2 krążowniki
- 1 niszczyciel

Uwaga 1! Oczywiście, w idealnym świecie powinniśmy założyć ograniczenia na ustawianie pól statków (np. pola mogą być tylko obok siebie, statek musi być w całości położony pionowo lub poziomo na planszy, statki nie mogą się stykać i przekrywać, itp.) . W naszym prostym programie możemy pominąć tego typu zabezpieczenia. Oczywiście jeśli komuś starczy czasu to można próbować to implementować.

Uwaga 2! Na stronie znajduje się przykładowy skompilowany program (bez wyżej wspomnianych zabezpieczeń).

Punktacja:

1. Stworzenie menu głównego wraz z opcją wyjścia (**1 p.**)
2. Obsłużenie opcji „Nowa gra” wraz ze stworzeniem struktury `Punkt` i narysowaniem planszy (**9 p.**)
3. Makefile (**dodatkowo 1 p.**)

Przykład piku `Makefile`

```
CC=gcc
CFLAGS=-Wall -pedantic -std=c99
LIBS=-lm

all: program

program: main.o struktura1.o struktura2.o
	$(CC) -o main main.o struktura1.o struktura2.o $(CFLAGS) $(LIBS)

struktura1.o: struktura1.c struktura1.h struktura2.h
	$(CC) -o struktura1.o -c struktura1.c $(CFLAGS)

struktura2.o: struktura2.c struktura2.h struktura1.h
	$(CC) -o struktura2.o -c struktura2.c $(CFLAGS)

main.o: main.c struktura1.h struktura2.h
	$(CC) -o main.o -c main.c $(CFLAGS)

clean:
	rm -rf *.o main
```

Jak to działa? Tworzymy plik tekstowy o nazwie **Makefile** i tworzymy jego zawartość na podstawie tego co wyżej. **Uwaga!** Odstępy w linijkach (np. zaczynających się od `$(CC)` itd.) to **tabulacje** (nie spacje) – użycie Tab jest niezbędne!

Następnie zamiast kompilować wpisując całą komendę w linii poleceń, wpisujemy tylko „make”.

Uwaga! Makefile nie wyłapuje zmian w plikach .h, dlatego najlepiej zrobić:

```
make clean
make
```

Komenda `make clean` usunie wcześniej skompilowane pliki.