Instrukcja projektowa; projekt numer 1

Podstawy Programowania 2017/18, kierunek Informatyka + Inżynieria Danych

autor: Dariusz Dereniowski[1]

wersja z dnia: 13.11.2017 r.

Projekt Puzzle Binarne

Cel

Celem projektu jest implementacja jednoosobowej gry puzzle binarne w postaci programu **konsolowego**. Zasady gry są następujące. Gra odbywa się na planszy o wymiarach *N*x*N* (*N* jest parzyste). W każdym polu planszy może znajdować się cyfra 0 lub 1. Plansza jest wypełniana przez gracza przy zachowaniu następujących reguł:
(1) W żadnej kolumnie i żadnym wierszu nie wysępują obok siebie więcej niż dwie identyczne cyfry.
(2) W każdym wierszu i kolumnie znajduje się tyle samo cyfr 0 i 1 (tzn. jest *N*/2 cyfr każdego rodzaju).
(3) Nie występują dwa identyczne wiersze ani dwie identyczne kolumny.
Gracz może wpisywać cyfry tylko w pola, które nie były początkowo wypełnione (pola te będziemy nazywać *modyfikowalnymi*).

	0		0					1	
	0			0			1		
			0	0		0			
0			0		0	0			
							0		
								1	1
1		1	0			1			
	0		0	1				1	
					1		0		

Ilustracja 1: (Źródło: http://binarypuzzle.com/)

Więcej szczegółów na temat gry można znaleźc na stronie:
`http://binarypuzzle.com/`

Ogólne wytyczne

Za projekt można uzyskać 0-18 punktów. 15 punktów stanowi tzw. 100% za to zadanie projektowe, a pozostałe 3 punkty to wymagania ponadprogramowe ("z gwiazdką"), przy czym 5 punktów to wymagania obowiązkowe. Realizacja wymagań obowiązkowych konieczna jest aby projekt był oceniany.

Program powinien być napisany z użyciem udotępnionego szablonu. Szablon umożliwia uzyskanie zaawansowanych możliwości w zakresie obsługi terminala w systemie Windows. **Zabronione jest**

1 Uwaga: W razie niejasności lub niejednoznaczności w poniższym opisie proszę kontaktować się z autorem instrukcji; pokój EA 209

używanie instrukcji cin/cout/printf/scanf. W celu pisania na ekran i czytania z klawiatury, należy używać wyłącznie metod dostępnych w szablonie. **Uwaga**: udostępniony szablon działa w systemie Windows i nie ma możliwości łatwego przeniesienia go do innych systemów operacyjnych. Projekt należy zatem przygotować **wyłącznie dla systemu Windows** z wykorzystaniem szablonu.

Pamiętaj o przygotowaniu się do odbioru - koniecznie sprawdź wcześniej czy potrafisz sprawnie uruchomić swój program na komputerze w laboratorium!

Obsługa programu

Program powinien wykorzystywać klawiaturę w następujący sposób (zarówno duże jak i małe litery):

strzałki	przemieszczanie się kursorem po polach planszy
esc	wyjście z programu
n	nowa gra
01	wpisanie cyfry na planszę
o	losowe wypełnianie planszy
p	prosta podpowiedź
r	zmiana rozmiaru planszy
k	proste sprawdzanie możliwości ukończenia gry
d	sprawdzenie reguły (2)
a	automatyczna detekcja końca gry
j	podświetlenie „jednoznacznych” pól
w	wypełnienie podświetlonych „jednoznacznych” pól
s	zapisanie stanu gry
l	odczytanie stanu gry
b	pełne sprawdzenie i pokazanie przykładowego wypełnienia pól

Wymagania obowiązkowe (5 pkt.)

Wszystkie wymienione tutaj elementy należy zaimplementować. Brak któregokolwiek z poniższych elementów skutkuje otrzymaniem 0 pkt. z tego projektu.

(a) Wyświetlanie planszy oraz legendy programu. Plansza powinna posiadać ramkę. Legenda powinna zawierać imię, nazwisko oraz numer indeksu autora, listę zaimplementowanych funkcjonalności (w postaci wymienienia liter punktów z niniejszej instrukcji) oraz listy działających skrótów klawiszowych. Domyślnie legenda powinna być wyświetlona z lewej strony ekranu, natomiast plansza po prawej. Powinna być możliwość łatwego przesunięcia tych elementów poprzez zmianę odpowiednich stałych w kodzie programu.

(b) Poruszanie się po planszy z użyciem kursorów. W ramach legendy powinna być wyświetlona bieżąca pozycja kursora. Obsługa klawisza *esc* kończącego program.

(c) Możliwość wpisywania cyfr na planszy. Wciśnięcie klawisza 0 lub 1 powinno skutkować pojawieniem się tej cyfry na planszy na pozycji wskazywanej przez kursor, o ile pole jest modyfikowalne. Cyfry wpisane przez użytkownika powinny być wyróżnione w stosunku do cyfr znajdujących się początkowo na planszy (pobranych zgodnie z punktem (e) poniżej).

(d) Sprawdzanie reguły (1) podanej w zasadach gry. Sprawdzenie skutkuje tym, że jeśli wpisanie cyfry przeczyłoby regule (1), to cyfra ta nie jest umieszczana na planszy.

(e) Nowa gra. Wciśnięcie klawisza *n* powoduje rozpoczęcie nowej gry z domyślną planszą:

```
...1.......1
.0......1...
.....0......
.0.1.0.0.0.0
1.1....0..1.
.....0..1...
..11....1...
0.....0....0
.1..1.....1.
..0..0.0....
..0..00....0
...1..0..1..
```

Wymagania nieobowiązkowe (10 pkt.)

(f) (**1 pkt.**) Sprawdzanie reguł (2) oraz (3). Sprawdzenie skutkuje tym, że jeśli wpisanie cyfry na polu (modyfikowalnym) wskazywanym przez kursor skutkowałoby pogwałceniem reguły (2) lub (3), to cyfra ta nie jest umieszczana na planszy. **Uwaga:** ten punkt należy zaimplementować, aby otrzymać punkty za funkcjonalności opisane dalej.

(g) (**1 pkt.**) Losowe częściowe wypełnienie planszy. Wciśnięcie klawisza *o* powoduje, że pewna, losowo wybrana, liczba pól zostanie wypełniona losowymi wartościami, przy czym sposób wypełnienia powinien być poprawny, tzn. nie przeczyć żadnej z reguł gry. Proponowany sposób wypełnienia, to:

- rozpocznij od pustej planszy
- losuj pole oraz wartość cyfry
- jeśli możliwe jest wpisanie danej cyfry w wylosowane pole, to wpisz tą wartość
- jeśli wylosowane wartości nie są poprawne, to je porzuć i przejdź do losowania następnych wartości
- zakończ wypełnianie gdy osiągnięto wypełnienie wymaganej liczby pól lub gdy liczba prób przekroczyła określony w programie próg. Zarówno oczekiwana liczba wypełnionych pól jak i próg powinny być określone stałymi, które jest łatwo w programie zmienić. Sugeruje się oczekiwaną liczbę pul określić dwiema stałymi, które określają maksymalne procentowe wypełnienie planszy oraz minimalne procentowe wypełnieni planszy i losowość polega na wybraniu liczby *x* pomiędzy tymi dwiema wartościami, co skutkuje tym, ze *x*% pól planszy zostanie wypełnionych tym algorytmem. Maksymalna liczba prób też powinna być uzależniona od rozmiaru planszy (im większa plansza tym więcej prób podejmujemy - na przykład liczba pól planszy do kwadratu itp.)
- wylosowane powyżej pola powinny stać się polami, które nie są modyfikowalne (innymi słowy, punkt ten realizuje rozpoczęcie nowej rozgrywki, ale nie od planszy domyślnej, lecz losowo wybranej).

Uwaga: ten punkt należy zaimplementować, aby otrzymać punkty za funkcjonalności opisane dalej.

(h) (**1 pkt.**) Prosta podpowiedź. Wciśnięcie klawisza *p* powoduje, że obok legendy wyświetlona jest informacja o tym jakie cyfry są dostępne dla pola, na którym znajduje się kursor.

Oczywiście jeśli pole jest niemodyfikowalne, to program informuje o tym fakcie i wpisanie żadnej cyfry nie jest wtedy możliwe. Jeśli dana cyfra jest niepoprawna dla danego pola, to powinna (w postaci krótkich komunikatów) zostać podana kompletna lista powodów uniemożliwiających umieszczenie cyfry w tym polu (przez powody rozumiemy wszystkie reguły wraz ze wskazaniem dalszych szczegółów, zgodnie z wymaganiami danej reguły – np. „Wpisanie cyfry 0 spowoduje, że wiersz będzie identyczny z wierszem numer *x*."). **Uwaga:** ten punkt należy zaimplementować, aby otrzymać punkty za funkcjonalności opisane dalej.

(i) (**2pkt.**) Ustalanie rozmiaru planszy przez użytkownika. Po wciśnięciu przycisku *r* użytkownik ma możliwość podania wartości *N*, czyli rozmiaru planszy. Zmiana rozmiaru planszy powoduje rozpoczęcie gry od początku. Należy zaimplementować dynamiczne przydzielanie pamięci, tzn. program powinien dynamicznie przydzielić pamięć na wszelkie tablice, których rozmiary zależą od wartości podanej przez użytkownika (jak na przykład tablica przechowująca wartości 0/1 na planszy). Minimalny rozmiar planszy wynosi 2, natomiast maksymalny rozmiar jest określony przez dostępną przestrzeń na ekranie. (Uwaga: pomimo tego, że maksymalny rozmiar tablicy jest również możliwy do oszacowania na etapie pisania kodu, należy korzystać z dynamicznego przydzielania pamięci.) Zmiana rozmiaru planszy rozpoczyna nową rozgrywkę z następującym wypełnieniem planszy:
   - jeśli w katalogu bieżącym znajduje się plik z wypełnieniem początkowym dla danego rozmiaru planszy, to wypełnienie początkowe jest pobierane z tego pliku. Należy dobrać samodzielnie nazewnictwo plików aby rozpoznawać istnienie lub brak dla danych rozmiarów, np. plansza8x8.txt, plansza4x4.txt itd.
   - w przypadku braku pliku wypełnienie jest losowane tak jak w punkcie (g).

(j) (**1 pkt.**) Proste sprawdzenie możliwości ukończenia gry. Po wciśnięciu klawisza *k*, program oznacza wszystkie pola, które są puste (tzn. nie zawierają żadnej cyfry) i których nie można uzupełnić żadną z cyfr (bezpośrednio sprawdzając reguły (1)-(3) dla każdego pola). Po naciśnięciu jakiegokolwiek klawisza zaznaczenie tych pól jest usuwane z planszy. Samo sprawdzenie nie kończy gry, tzn. grę można kontynuować.

(k) (**1 pkt.**) Dodatkowa podpowiedź dotycząca reguły (2). Przy każdym wierszu i każdej kolumnie wyświetlane są liczby zer i liczby jedynek w danym wierszu/kolumnie.

(l) (**1 pkt.**) Tryb automatycznej detekcji końca gry. Wciśnięcie przycisku *a* powoduje, że po każdym wpisaniu cyfry na planszy nastąpi sprawdzenie warunków podanych w punktach (j) oraz (k). Jeśli się okaże, że gry nie można ukończyć, to wyświetlany jest odpowiedni komunikat. Ponowne wciśnięcie klawisza *a* wyłącza ten tryb automatyczny. Fakt, czy tryb automatyczny jest włączony należy zaznaczyć wyświetlaniem odpowiedniego komunikatu na ekranie.

(m) (**1 pkt.**) Podświetlenie jednoznacznych pól. Po naciśnięciu klawisza *j* program zaznacza (innym kolorem) na planszy wszystkie pola, które można wypełnić tylko jedną wartością. Wciśnięcie klawisza *w*, gdy pola te są podświetlone, powoduje wpisanie tych jednoznacznych wartości w te pola. Wciśnięcie jakiegokolwiek innego klawisza powoduje usunięcie podświetlenia.

(n) (**1 pkt.**) Zapis i odczyt stanu gry. Wciśnięcie klawisza *s* powoduje zapisanie stanu gry do pliku o nazwie podanej przez użytkownika. Wciśnięcie pliku *l* powoduje odczyt stanu gry z pliku podanego przez użytkownika. Zapis/odczyt stanu powinny uwzględniać nie tylko sytuację na planszy, ale również ewentualne podświetlenia lub komunikaty jeśli są w danym momencie widoczne.

Wymagania „z gwiazdką” (3 pkt.)

Uwaga: poniższe wymagania są oceniane wyłącznie gdy zaimplementowane zostały wszystkie poprzednie punkty (a)-(n).

(o) (**3 pkt.**) Implementacja pełnego sprawdzenia czy możliwe jest dokończenie trwającej rozgrywki. Użytkownik po wciśnięciu klawisza *b* otrzymuje komunikat, który informuje czy istnieje możliwość poprawnego wypełnienia pustych pól. Ponowne wciśnięcie klawisza *b* powoduje pokazanie przykładowego sposobu poprawnego wypełnienia pustych pól, o ile takowe istnieje – te wartości należy wyróżnić innym kolorem, aby można było je odróżnić od dotychczas znajdujących się wartości na planszy. Bez względu na to czy użytkownik wcisnął klawisz *b* ponownie czy nie, dowolny inny klawisz powoduje kontynuację dotychczasowej gry.

Uwagi końcowe

- Konfiguracja programu powinna umożliwiać łatwą zmianę wszelkich parametrów, nie tylko tych wyraźnie wskazanych w powyższym opisie. Przez łatwą zmianę rozumiemy modyfikację stałej w programie.
- Projekt może być napisany w sposób obiektowy, ale całkowicie zabronione jest używanie biblioteki standardowej C++ (w tym typu string, cin, cout, vector itp.) (Uwaga: typu string z biblioteki C++ nie należy mylić w biblioteką string.h z C – można używać funkcje znajdujące się w string.h)
- Obsługa plików powinna być zrealizowana przy użyciu biblioteki standardowej C (rodzina funkcji f???? - np. fopen, fread, fclose itd.) Nie można w tym celu używać mechanizmów C++ (np. fstream).
- Każdy fragment przedstawionego do oceny kodu powinien być napisany samodzielnie przez studenta. Nie jest dozwolone korzystanie z kodu znalezionego w Internecie, otrzymanego od innych osób lub napisanego z pomocą innych osób (z wyłączeniem pomocy uzyskanej na konsultacjach projektowych).
- Należy zwrócić uwagę na właściwy podział kodu na funkcje w celu unikania powielania kodu. Na przykład może się okazać, że w kilku wskazanych punktach przydatny będzie kod, który sprawdza czy zadana cyfra spełnia warunki wybranej reguły gry w zadanym polu planszy. W takim wypadku naturalne i wskazane jest umieszczenie takiego kodu wewnątrz funkcji wywoływanej następnie we właściwych miejscach.
- Stałe jednostki w programie powinny być opisane odpowiednimi komentarzami, na przykład:

```
const double PREDKOSC_MARIO = 40.0;   // piksele na sekundę
const int SZEROKOSC_PLANSZY = 320;    // piksele
const double POZYCJA_X_NAPISU = 60.0; // procent szerokości ekranu
const double POZYCJA_Y_NAPISU = 5.0;  // procent wysokości ekranu
```