Instrukcja do zajęć laboratoryjnych

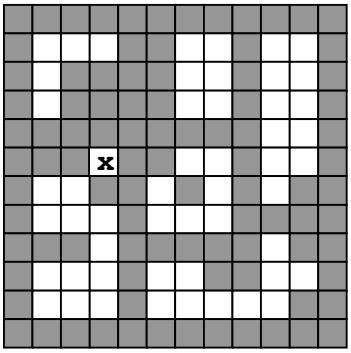
Język ANSI C (w systemie LINUX)

wersja: 1.2

Nr ćwiczenia:	10, 11	
Temat:	**Powierzchnia obszarów**	
Cel ćwiczenia:	Celem ćwiczenia jest napisanie programu umożliwiającego wyznaczenie: • liczby białych obszarów, • wielkość każdego białego obszaru. Obszary (patrz rysunek poniżej) zdefiniowane są w macierzy kwadratowej.	
Wymagane przygotowanie teoretyczne:	Szczegółowy opis zadania zamieszczony poniżej. Idea programów (algorytmów) rekurencyjnych.	
Sposób zaliczenia:	Sprawozdanie w formie pisemnej. Pozytywna ocena ćwiczenia przez prowadzącego pod koniec zajęć.	[**X**] []

1. Opis zadania

Napisać program umożliwiający obliczenie liczby białych obszarów i ich rozmiaru zdefiniowanych w kwadratowym obszarze o rozmiarze `N x N`. Przykładowe zadanie (pokazane w postaci rysunku) wygląda następująco:



Dla powyższych danych program powinien zwrócić następującą odpowiedź:

Liczba białych obszarów: 7

Rozmiar(y) białych obszarów: 1, 5, 6, 7, 10, 11, 12

2. Uwagi

- Pole „stykające” się tylko narożnikiem z obszarem nie należy do tego obszaru. Przykładowo pole oznaczone znakiem ‘x’ na rysunku, należy traktować jako oddzielny obszar o rozmiarze 1.

- Dla ułatwienia zadania zakładamy istnienie czarnego „obramowania” obszaru roboczego. Stad rzeczywisty obszar roboczy ma rozmiar `(N+2) x (N+2)` (zabieg taki uprości nam implementację w postaci algorytmu rekurencyjnego).

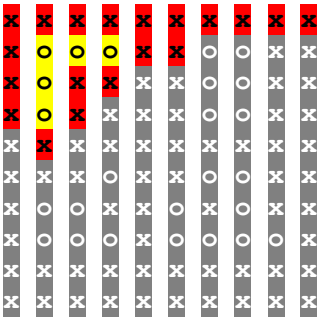
- Przechodź kwadrat wierszami i dla pierwszego nie odwiedzonego pola wywołuj funkcję przetwarzającą jeden obszar. Sztuka polega na użyciu czterech rekurencyjnych wywołań tej funkcji dla każdego nie odwiedzonego białego pola i oznaczeniu go specjalnym symbolem jako odwiedzonego (policzonego). (ustanowienie obszaru roboczego o rozmiarze `(N+2) x (N+2)` ułatwia implementację takiego właśnie algorytmu).

- Sposób podawania danych dla zadania można przyjąć dowolny (np. tablica zer i jedynek predefiniowana wewnątrz programu, odczyt danych o ustalonym formacie z pliku, „odczyt” pliku graficznego w postaci np. bitmapy, itp.).

- Program musi umożliwiać graficzną (najprościej w postaci „grafiki” tekstowej) wizualizację zadania. Innymi słowy na ekranie powinien pojawić się rysunek jak powyżej. Czarne pola można wyświetlić jako znaki **x**, białe jako znaki **o**. Pokazano to poniżej:

```
x x x x x x x x x x x x
x o o o x x o o x o o x
x o x x x x o o x o o x
x o x x x x o o x o o x
x x x x x x x x x o o x
itd...
```

W wersji bardzie złożonej (i preferowanej) wizualizacja zadania powinna polegać na tym, aby widać było, w jaki sposób rekurencja przeszukuje zadany obszar. Chodzi o wprowadzenie do programu pewnego opóźnienia czasowego (np. 0,5 sek.) oraz zaznaczanie na ekranie, który element tablicy jest w danym momencie analizowany. Warto rozważyć również użycie koloru. Na rysunku poniżej pokazano przykładowe rozwiązanie. Na początku obszar zaznaczony jest kolorem szarym. Każdy odwiedzony element zamieniany jest na kolor czerwony (dzięki wprowadzonemu opóźnieniu czasowemu widać to *on-line* na ekranie) a elementy poszczególnych obszarów na kolor żółty. W ten sposób program nabiera również pewnych walorów dydaktycznych, bo pozwala „naocznie” poznać działanie rekurencji. Po skończonym przeszukiwaniu cały obszar będzie pokolorowany na czerwono i żółto (bo każdy element musi być przynajmniej raz odwiedzony).



- Zadanie pochodzi z rozdziału 7 książki: *Struktury danych w języku C*, Adam Drozdek, Donald L. Simon, WNT, 1996. Tam też znaleźć można wiele cennych uwag i przykładów dotyczących zastosowań rekurencji w programach – również uwagi dotyczące powyższego zadania laboratoryjnego.

3. Sprawozdanie

Sprawozdanie powinno zawierać następujące elementy:

- wydrukowany kod źródłowy programu (do wydruku użyć koniecznie takiej czcionki: `Courier New 8pt`). Plik źródłowy musi być **rozsądnie skomentowany** oraz **czytelnie sformatowany** (możesz użyć jakiegoś programu narzędziowego, który pomoże ładnie sformatować tekst źródłowy). Nieczytelny kod nie będzie sprawdzany !

- plik *makefile* do kompilacji programu,

- dyskusję rozwiązania i rzeczowe uwagi autora programu.