do zdobycia: [20pkt]
czas realizacji: 2 tygodnie

Zestaw 4

Tym razem przyjrzymy się już problemowi bardziej złożonemu. Zadaniem jest stworzenie programu służącego do przetwarzania bitmap. Aby nie komplikować sobie życia za bardzo, do obsługi bitmap wykorzystamy gotową bibliotekę, mianowicie bibliotekę `EasyBMP`. Więcej informacji dotyczących tej biblioteki znaleźć można pod adresem:

`http://easybmp.sourceforge.net/`

Niezbędny przy realizacji tego ćwiczenia zasoby (źródła i dokumentację biblioteki `EasyBMP`) można także pobrać z adresu:

`http://www.mif.pg.gda.pl/homepages/swinczew/AR/EasyBMP.zip`

Na wstępie nasze zadanie nie będzie specjalnie złożone. Celem jest napisanie programu równoległego realizującego proste przetwarzanie bitmapy. Program ten powinien:

1. wczytać wskazany przez użytkownika (wektor argumentów) plik bitmapy,
2. obliczyć średnią luminancję bitmapy, luminancja pojedynczego piksela dana jest wyrażeniem:
$$Y = 0.299R + 0.587G + 0.114B,$$
gdzie symbole R, G i B oznaczają wartość trzech składowych koloru danego piksela,
3. przekonwertować bitmapę na czarno-białą (2-bitową, zachowując rozdzielczość), uzależniając wynikowy kolor (czarny lub biały) piksela od wejściowej luminancji piksela, pikselom o luminancji większej od średniej luminancji bitmapy powinien zostać przypisany kolor biały, zaś pozostałym pikselom kolor czarny,
4. zapisać bitmapę do pliku o nazwie określonej przez użytkownika (wektor argumentów).

Jako iż program ten będziemy rozwijać przez najbliższe parę tygodni (będzie on podstawą do realizacji kolejnego ćwiczenia), warto od samego początku zadbać, by był on napisany porządnie. Warto zatem wyodrębnić funkcje służące do:

1. odczytu bitmapy z pliku,
2. komunikowania bitmapy,
3. obliczania luminancji pojedynczego piksela,
4. obliczania średniej luminancji bitmapy,
5. przetwarzania bitmapy,
6. zapisu bitmapy do pliku.

Warto również stworzyć własne typy służące do przechowywania informacji o pojedynczym pikselu, jak i całej bitmapie. Tworząc te typy należy mieć na uwadze fakt, iż w przypadku przetwarzania równoległego zachodzi konieczność komunikowania (pomiędzy procesami) informacji o bitmapie (zastosowanie obiektów nie jest zatem wyborem specjalnie szczęśliwym).

Powstały program powinien:

1. wczytać (na masterze) bitmapę z pliku,
2. rozesłać (z mastera) informację o bitmapie do wszystkich procesów,
3. w sposób równoległy obliczyć średnią luminację bitmapy,
4. rozesłać informację o średniej luminancji do wszystkich procesów,
5. w sposób równoległy przekonwertować bitmapę na czarno-białą,
6. zebrać (do mastera) informację o przetworzonej bitmapie,
7. zapisać (na masterze) przetworzoną bitmapę do pliku.

Zakładamy, że jedynie nadzorca jest w stanie dokonać odczytu i zapisu bitmapy. W dekompozycji problemu (już dwuwymiarowego) można zastosować (chociażby) podejście typu mozaika (wyjaśnienie na tablicy). Należy zadbać, by zastosowana metoda dekompozycji problemu gwarantowała optymalny podział pracy. Również należy zadbać, by sposób komunikacji był możliwie jak najbardziej optymalny (minimalna liczba operacji komunikacji, wykorzystanie metod komunikacji zbiorowej).