# AAL - projekt: zalecenia ogólne

Proponowane problemy algorytmiczne są do opracowania *indywidualnego*. Może się zdarzyć, że to samo zadanie projektowe jest proponowane 2 lub 3 studentom w celu porównania różnych metod rozwiązania; w takim przypadku dodatkowe szczegółowe wymagania i oczekiwania zostaną przekazane takiemu zespołowi. Opracowywane algorytmy i realizujące je programy powinny spełniać następujące ogólne kryteria.

***Język programowania.*** Projekty powinny być opracowane w **standardowym C++**. Z biblioteki standardowej można korzystać bez ograniczeń; trzeba natomiast zaniechać używania rozszerzeń dostępnych lokalnie (na przykład bibliotek `<curses.h>` lub `<conio.h>`). Program powinien być przenośny i dać się skompilować/wykonać zarówno na platformie Linux jak i Windows (Visual C++ 2008). Oprócz standardowych akcesoriów C++ można skorzystać z biblioteki `<glut.h>`, wspieranej w miarę przenośnie na obydwu platformach.

***Samodokumentowanie.*** Każdy projekt powinien zawierać plik tekstowy `ReadMe.txt`, w którym należy umieścić zwięzłe informacje dotyczące organizacji kodu źródłowego oraz podstawowe dane o problemie i algorytmie. W szczególności powinna tam znaleźć się:

- "wizytówka" studenta
- krótka specyfikację problemu (albo przynajmniej tytuł problemu)
- opis konwencji dotyczących danych wejściowych i prezentacji wyników
- krótki opis metody rozwiązania, zastosowanych algorytmów i struktur danych
- informacje o funkcjonalnej dekompozycji programu na moduły źródłowe - nagłówkowe i implementacyjne ("przewodnik" po plikach źródłowych)
- inne informacje dodatkowe o szczególnych decyzjach projektowych (np. ograniczenia dotyczące rozmiaru problemu, charakterystyki generatorów danych testowych, specjalne konwencje w alokacji pamięci dynamicznej, wymagania dotyczące typów parametryzujących szablony, konwencje związane z obsługą sytuacji wyjątkowych, itp.).

Plik `ReadMe.txt` nie zastępuje dokumentacji końcowej!
Każdy plik źródłowy musi być opatrzony komentarzem nagłówkowym z nazwiskiem studenta i tytułem problemu.

***Dekompozycja.*** Program składa się z dwu logicznych części: części realizującej struktury danych i algorytmy potrzebne w rozwiązaniu problemu (definicja i implementacja odpowiednich klas) oraz z części realizującej interfejs użytkownika. Projektując warstwę algorytmiczną należy oczywiście brać pod uwagę potrzeby interakcji z użytkownikiem i stosowane konwencje wejścia-wyjścia, ale nie powinno to powodować sztywnego związania z tymi konwencjami (np. zmiana konwencji we-wy z tekstowej na graficzną nie powinna wymagać modyfikacji warstwy algorytmicznej a jedynie modyfikację/uzupełnienie warstwy odpowiedzialnej za interfejs użytkownika).

***Samoprezentacja programu.*** Program powinien zachowywać się bez tajemnic użytkowych. W szczególności jeden z typowych sposobów aktywacji programu (np. wg konwencji systemu UNIX) powinien pokazywać wszystkie prawidłowe formy aktywacji lub podawać nazwę pliku z informacjami dodatkowymi.

***Wybór konwencji we-wy.*** Programy opracowywane są zasadniczo dla we-wy znakowego. Jest ważne, aby postać danych i wyników była możliwie prosta i czytelna dla użytkownika. W niektórych przypadkach, przy spełnieniu pewnych ograniczeń, można wyniki prezentować w postaci pseudograficznej. Dla niektórych problemów warto rozważyć możliwość prezentacji graficznej z użyciem biblioteki `glut.h`.

***Dane wejściowe (instancje problemu)***. Wybór danych do testowania algorytmu powinien uwzględniać szczególne sytuacje istotne z punktu widzenia (1) badania poprawności (2) oceny złożoności. W większości przypadków generacja instancji problemów wg żądanych rozmiarów jest zagadnieniem samym w sobie i wymaga opracowania pomocniczych algorytmów generacji. Jeżeli dwa lub więcej projektów korzysta z tego samego sposobu reprezentacji danych wejściowych, to może powstać wspólny generator, jako oddzielna aplikacja (ewentualnie parametryzowana). Bezwzględnie prosimy unikać „zaszywania" w kodzie nazw plików z danymi wejściowymi czy z wynikami i innych usztywniających chwytów (np. umieszczania w danych statycznych instancji testowych problemu). Podczas uruchamiania i testowania można pozwoliś sobie na większą swobodę, ale ostateczna wersja powinna być pod tym względem „czysta".
***Tryby wykonania.*** Program powinien umożliwiać 3 rodzaje wykonań:

1. wg danych dostarczonych ze strumienia wejściowego (standardowego lub pliku) dla sekwencji konkretnych problemów
2. wg danych generowanych automatycznie (ewentualną parametryzację generacji określa użytkownik)
3. wykonanie z generacją danych, pomiarem czasu i prezentacją wyników pomiarów.

Dla niektórych problemów mogą być celowe inne tryby wykonania (do ustalenia z prowadzącym).

***Styl.*** W programie należy przestrzegać zasady dobrego stylu kodowania (przejrzysty układ tekstu programu uwzględniający konstrukcje zagnieżdżone; odpowiednia zawartość i porządek konstrukcji językowych w plikach nagłówkowych i plikach implementacyjnych; stosowanie identyfikatorów ułatwiających zrozumienie kodu; zwięzłe komentarze dla funkcji nietrywialnych; komentarze objaśniające algorytm wewnątrz funkcji). Nazwy plików powinny sugerować ich zawartość i rolę w programie.

Przed przekazaniem projektu do oceny proszę przeczytać kod i usunąć pozostające zwykle po fazie uruchamiania rozmaite nieużytki, dziwne komentarze itp. Proszę także zatroszczyć się o uzyskanie kodu z czystą kompilacją (bez ostrzeżeń).

***Dokumentacja.*** Wraz z programem źródłowym trzeba przekazać dokumentację końcową (dokument *Word* lub *pdf*) zawierającą opis problemu i metody (lub metod) rozwiązania, opis wykorzystywanych struktur danych i algorytmów pomocniczych oraz ocenę spodziewanej złożoności algorytmu. Prostą wskazówkę, którą należy kierować się przygotowując dokumentację programu, zawiera pytanie: *czy opis zadowoliłby mnie samego gdybym o problemie i jego rozwiązaniu miał dowiedzieć się z tego dokumentu?* Ewentualne dodatkowe ustalenia szczegółowe dotyczące zawartości dokumentacji końcowej należy uzgodnić z prowadzącym projekt.

***Pomiary czasu wykonania.*** W każdym projekcie oczekuje się: (1) przeprowadzenia analizy złożoności zaproponowanego algorytmu oraz (2) wsparcia dla wykonywania eksperymentów z pomiarami czasu dla różnych (wybranych przez użytkownika) rozmiarów problemu. Wynikiem analizy jest oszacowanie asymptoty $O(T(n))$; wsparcie dla eksperymentów pomiarowych polega na możliwości rejestracji ciągu wartości $t(n_1), \ldots t(n_k)$ konkretnych czasów wykonania dla instancji problemu o rozmiarach $n_1, \ldots, n_k$ i **generacji zestawienia wyników jak w poniższej tabeli**. Jeśli ocena teoretyczna $T(n)$ jest zgodna z wynikami pomiarów, to potwierdzenie tego powinno być natychmiast widoczne w tabeli.

| Algorytm z asymptotą $O(T(n))$ | | |
|---|---|---|
| $n$ | $t(n)[ms]$ | $q(n)$ |
| $n_1$ | $t(n_1)$ | . |
| $n_2$ | $t(n_2)$ | . |
| .... | .... | . |
| $n_{mediana}$ | $t(n_{mediana})$ | *1.0* |
| .... | .... | . |
| $n_{k-1}$ | $t(n_{k-1})$ | . |
| $n_k$ | $t(n_k)$ | . |

Przez $q(n)$ oznaczono (znormalizowany dla $n_{mediana}$ ) współczynnik zgodności oceny teoretycznej z pomiarem czasu:

$$q(n) = \frac{t(n)}{cT(n)} = \frac{t(n)\ T(n_{mediana})}{T(n)\ t(n_{mediana})}$$

Według takiej tabelki łatwo ocenić, czy ocena teoretyczna jest dobra (w kolumnie $q(n)$ wszystkie wartości bliskie 1), czy też w ocenie $T(n)$ nastąpiło przeszacowanie ($q(n) < 1$) bądź niedoszacowanie ($q(n) > 1$). Użytkownik powinien móc wybrać rozmiary problemu, dla których zostaną przeprowadzone pomiary. Pomiary czasu powinny korzystać ze standardowych, przenośnych, akcesoriów biblioteki C++ (funkcja `clock()` z biblioteki `<ctime>`).

***Terminy.*** Do dnia **19 kwietnia 2012** trzeba przygotować koncepcję rozwiązania problemu i przedstawić ją prowadzącemu projekt do zaakceptowania. Niedotrzymanie terminu może spowodować zmniejszenie liczby punktów za projekt (**do 5p za tydzień opóźnienia**). Zakończenie projektu (prezentacja ostatecznej wersji algorytmu i przekazanie dokumentacji końcowej) przypada na dzień **31 maja 2012; termin 14 czerwca przeznaczony jest na ewentualne uzupełnienia / korekty projektu według ustalenia z prowadzącym.**

Konsultacje dotyczące projektu przewidziane są w **czwartki od 14:15** lub w innych terminach uzgodnionych indywidualnie z prowadzącym.

A.Pająk