Лекция 2 (11. Октомври, 2011)

Summary
На първата лекция тази година бяха разгледани съвсем уводни въпроси от програмирането. През първия час бяха показани няколко хубави практики за писане на по-четлив и организиран код. Беше споменато как да го организираме в отделни .h и .cpp файлове, и каква е ползвата от това да пишем проекта си на „модули" – тоест отделни парчета, които комуникират, но не зависят едно от друго.

Първи час
Часът започна с демонстрация какви са най-честите проблеми в кода на начинаещите програмисти, а именно:

- Индентация
- Лош избор на имена на променливи
- Неконсистентен стил на писане
- Произволно позициониране на отварящи и затварящи скоби
- Писане на повече от един логически statement на един ред
- Неенкапсулирани данни

Като прост пример беше даден следният програмен фрагмент (като считаме, че има общо 100 студента, разделени в 4 групи, като във всяка от тях има поне по 1 студент):

```
int Ocenki[100];
int Grupa[100];
int sum1 = 0, sum2 = 0, sum3 = 0, sum4 = 0, cnt1 = 0, cnt2 = 0, cnt3 =
0, cnt4 = 0;
for (int i = 0; i < 100; i++)
{if (Grupa[i] == 0) {sum1 += Ocenki[i]; cnt1++;}
if (Grupa[i] == 1) {sum2 += Ocenki[i]; cnt2++;}
if (Grupa[i] == 2) {sum3 += Ocenki[i]; cnt3++;}
if (Grupa[i] == 3) {sum4 += Ocenki[i]; cnt4++;}}
cout << "Srednata ocenka na grupa 1 e: " << sum1 / cnt1 << endl;
cout << "Srednata ocenka na grupa 2 e: " << sum2 / cnt2 << endl;
cout << "Srednata ocenka na grupa 3 e: " << sum3 / cnt3 << endl;
cout << "Srednata ocenka na grupa 4 e: " << sum4 / cnt4 << endl;
```

който беше „трансформиран" до:

```
struct Student {
    int grade;
    int group;
};

const int NUM_STUDENTS = 100;
const int NUM_GROUPS = 4;

Student students[NUM_STUDENTS];
int studentCount[NUM_GROUPS] = {0};
```

```
int gradeSum[NUM_GROUPS] = {0};

for (int i = 0; i < NUM_STUDENTS; i++) {
    gradeSum[students[i].group] += students[i].grade;
    studentCount[students[i].group]++;
}

for (int i = 0; i < NUM_GROUPS; i++) {
    cout << "Mean grade in group " << i + 1 << " is: "
            << (double)gradeSum[i] / studentCount[i] << endl;
}
```

Беше наблегнато на следните неща:

- Да има индентация (1 табулация или фиксиран на брой шпации)
- Да се спазва някоя общоприета naming convention – или CamelCase или lowercase_separated_by_underscore, но не и двете заедно.
- Да не се смесват езици (в случая български с английски). За предпочитане е кодът да е на английски.
- Отделни смислови елементи (присвоявания, цикли, кондиции и т.н.) да бъдат на отделни редове.
- Къдравите скоби да отделят по подчертан начин даден блок код.
- Да се ползват значещи имена на променливите (като евентуално са позволени еднобуквени променливи ако, например, са итератори в цикъл, както в случая е i).

Беше само спомената ползата от използване на .h и .cpp файлове за всеки отделен модул (или клас), както защо е хубаво да кръщаваме тези файлове с отново значещи имена.
Надълго и нашироко беше обяснено защо е хубаво един голям проект да се състои от отделни, заменяеми модули пред това целият код да е изсипан на едно място и да зависи от общи ресурси. Въпреки потенциалната печалба откъм време и евентуално по-малко код, разбиването на проекта на независими парчета помага неимоверно за неговата поддръжка. Друго предимство е така нареченото „преизползване" на кода – тоест един модул би могъл да бъде използван в друг проект. Със същата идея е създадена и Standard Template Library (STL), която предоставя различни модули, реализиращи алгоритми и структури данни, които намират приложение в широка гама задачи.

Линкове:
http://en.wikipedia.org/wiki/Naming_convention_(programming)
http://geosoft.no/development/cppstyle.html
http://www.possibility.com/Cpp/CppCodingStandard.html
http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml

Втори час (задачи)

1. Крис имаше рожден ден! И всичко щеше да бъде перфектно, ако някой не беше изрязал правоъгълно парче от вътрешността на правоъгълната торта. На всичкото отгоре, изрязаното парче беше накриво, тоест страните му не бяха успоредни на страните на тортата! Оказа се, че всички гости вече са яли от тортата, освен Ели и Крис. Как обаче да си поделят тортата поравно? Да не говорим, че ножът е толкова изхабен, че може да направи най-много още един разрез (добре де, това е малко странно).
Чакайте! Не всичко е загубено! Може би има начин остатъкът от тортата да бъде разделен на 2 равни части с един разрез! Можете ли да кажете откъде трябва да минава той, ако има такъв?

Абстракция:
Даден е бял правоъгълник, във вътрешността на който е нарисуван друг, черен правоъгълник, чиито страни може да не са успоредни на тези на белия. Може ли с една единствена права линия да се раздели остатъка от белия правоъгълник на части с равна площ?

2. Даден е масив с N числа. Изградете алгоритъм, който намира негов подинтервал с максимална сума. Подинтервал е поредица от последователни негови елементи.

3. Ели се намира в стая, която има 2 врати: едната води към нейните най-големи кошмари (място, където ноктите й се чупят по-често, косата й не е толкова блестяща, не всички я харесват, и всеки ден има изпити по анализ), а другата - към нейния рай (всъщност не сме много сигурни какво е това, но бихме могли да предположим, че Станчо се намира там). Всяка от вратите се охранява от пазач, единият от които винаги казва истината, а другият - винаги казва лъжата. Ели има право на един единствен въпрос и след това трябва да избере врата, през която да мине. Какъв би могъл да бъде този въпрос?

4. Ели е с група от N-1 свои приятели. За да разнообразят иначе нормалните си занимания, те решават да си направят тройка. Както се досещате, тройка от три момичета или три момчета не би било едно от най-интересните неща в света. Затова на всеки от приятелите на Ели, и на нея самата, е определено едно число "мъжественост", като момчетата са с положителни числа (някои не толкова, даже стигащи до нула), а момичетата са с отрицателни числа (отново някои, за съжаление, достигат нула). За да е добре-балансирана една тройка, сумата на числата на тримата участника в нея трябва да е равна на нула. Ели се пита колко такива различни тройки съществуват?

Абстракция
Даден е масив с N числа. Изградете алгоритъм, който намира дали съществува

тройка негови числа със сума нула. Разгледайте решения със сложности O(N**3), O(N**2 * logN), O(N**2).