Języki Programowania – C++ – Zajęcia 8

Napisz program w C++, który:

1. **Jako argumenty wiersza poleceń przyjmuje:**
   - nazwę pliku z liczbami
   - ilość liczb które należy wczytać
   - opcjonalną operację: sum, avg, max, min, median
2. **Wczytuje liczby do obiektu klasy Analyzer.**
3. Klasa **Analyzer musi używać std::vector<double> do przechowywania danych**.
4. Klasa **Analyzer musi udostępniać** metody:
   - `void loadFromFile(const std::string& filename);`
   - `double print();` - wypisuje wszystkie liczby w jednym wierszu oddzielone spacjami
   - `double sum();`
   - `double average();`
   - `double max();`
   - `double min();`
   - `double median();` *(powinna sortować wektor)*
5. Konstruktor domyślny powinien tworzyć pusty zbiór danych.

6. Program główny (main) powinien:
   - odczytać argumenty, // 0.5 pkt
   - załadować dane do obiektu klasy Analyzer, wypisać metodą print //1 pkt
   - wywołać odpowiednią metodę, wypisać wynik. //2.5 pkt, po 0.5 za każdą metodę
7. Program powinien obsłużyć błędy: //0.5 pkt
   - zbyt mało argumentów,
   - nieistniejący plik,
   - pusty plik (lub zbyt mało liczb).
8. Należy ponadto zapoznać się z treścią pliku: http://www.if.pw.edu.pl/~majanik/data/JP/2012/makefile.pdf . Napisany Makefile, powinien umożliwiać kompilację napisanego programu, definiować zmienną CXX określającą kompilator (g++) oraz CXXFLAGS definiującą flagę (-Wall), a tworzone klasy powinny być wstępnie kompilowane do plików *.o. Należy tak skonfigurować Geany, by móc kompilować przez cegiełkę przy użyciu stworzonego makefile'a (wystarczy raz wybrać z rowzijanej listy przy cegiełce „Make all", by ta opcja wskoczyła jako domyślna). //0.5 pkt

```
Przykładowe wywołania
./analyzer 100 dane.txt sum
./analyzer 50 liczby.txt median
./analyzer 20 input.txt
Brak operacji ⇒ domyślnie wykonaj sum.
```

Informacje dodatkowe: Parametry wywołania programu (zmiany w deklaracji funkcji main):

```
int main(int argc, char **argv){
}
```

Gdzie:
argc - liczba parametrów, z którymi został wywołany program
**argv - tablica parametrów (każdy jako char*):
argv[0] - nazwa programu
argv[1], argv[2]... - kolejne parametry wywołania programu

Np. wywołanie programu program mogło by mieć formę:
./program Ala 3
Wtedy:
//argc == 4
//argv[0] == "program"
//argv[1] == "Ala"
//argv[2] == "3" || **zmiana "3" na 3 (liczbę) – funkcja atoi. np int a = atoi("3");** □
z biblioteki <stdlib.h>

Przyporządkowanie tych parametrów wykonuje się samoistnie w momencie wywołania programu, jedynym wkładem programisty jest zadeklarowanie funkcji main w formie int main(int argc, char **argv)