Dodatek A: Kod źródłowy programu `StarsParPloter`

```
/* Autor: Krzysztof Piotr Wójcik, 15.12.2009r
pliki wejściowe:

/roboczy/FourierOut.txt - plik wynikowy Cepheusa
/roboczy/Types.txt - trzy kolumny:
                 id, periodaov, typ wg ASAS (lub "-" jeśli nie jest znany)
*/


#include <iostream>          // | załączam biblioteki
#include <sstream>           // |
#include <fstream>           // |
#include <cmath>             // |
#include <vector>            // |
#include <cstdlib>           // |
#include <sys/stat.h>        // |
#include <sys/types.h>       // |


using namespace std;         // korzystam ze standardowej przestrzeni nazw

//---------------------------------------------------------------------------

string parValue(const int i)       // funkcja zwraca nazwę parametru...
  {
  switch (i)                       // zgodnie z porządkiem:
    {
    case 0: return "R21";
    case 1: return "R31";
    case 2: return "R41";
    case 3: return "phi21";
    case 4: return "phi31";
    case 5: return "phi41";
    }
  }

//---------------------------------------------------------------------------

void makeCommandsBaseR(ostream& out, const int i,const string& id)
// funkcja przekazuje do strumienia out podstawę instrukcji dla GNUplota
// do rysowania wykresów par. R i phi
  {
  string par = parValue(i);                          // sprawdzam nazwę par.
  out<< "set terminal jpeg giant size 800,640;\n"    // ustawiam plik outputu
      "set output \""<<id<<"/"<<par<<".jpg\" ;\n"    // nazywam plik wynikowy
      "set xlabel \"log (P[day])\";\n"               // opisuję oś x
      "set ylabel \" "<<par<<" \"; \n"               // opisuję oś y
      "set title \" "<<par<<" parameter vs log(P) \"; \n"
                                                     // nadaję wykresowi tytuł
      "set autoscale xy; \n"                         // skaluję osie
      "set key top left; \n"                         // ustawiam legendę
      "plot "
      "\"DCEP-FO.dat\" using (log($1)):"<<(2*i+20)<<":"<<(2*i+21)<<" with yer title \"DCEP-FO (ASAS)\" , "
      "\"DCEP-FU.dat\" using (log($1)):"<<(2*i+20)<<":"<<(2*i+21)<<" with yer title \"DCEP-FU (ASAS)\" , "
      "\"CW-FO.dat\" using (log($1)):"<<(2*i+20)<<":"<<(2*i+21)<<" with yer title \"CW-FO (ASAS)\" ";
                                                     // polecenie rysowania
  }

//---------------------------------------------------------------------------
```

```
void makeCommandsBaseRT(ostream& out, const string& id)
// funkcja przekazuje do strumienia out podstawę instrukcji dla GNUplota
// do rysowania wykresów parametru Rising Time
  {
  out<<     "set terminal jpeg giant size 800,640; \n"     // ustawiam plik outputu
      "set output \""<<id<<"/risingTime.jpg\"; \n"         // nazywam plik wynikowy
      "set xlabel \"log (P[day])\"; \n"                    // opisuję oś x
      "set ylabel \" rising time \"; \n"                   // opisuję oś y
      "set title \" rising time vs log(P) \"; \n"          // nadaję wkresowi tytuł
      "set autoscale xy; \n"                               // skaluję osie
      "set key bottom right; \n"                           // ustawiam legendę
      "plot \"DCEP-FO.dat\" using (log($1)):32 with points title \"DCEP-FO (ASAS)\" , "
           "\"DCEP-FU.dat\" using (log($1)):32 with points title \"DCEP-FU (ASAS)\" , "
           "\"CW-FO.dat\" using (log($1)):32 with points title \"CW-FO (ASAS)\" ";
                                                           // polecenie rysowania
  }

//-----------------------------------------------------------------------------

void tuPisz(ofstream& out, string& period, string& type, vector<string>& f)
// funkcja wypisuje dane do strumienia out
  {
  out<<period<<" ";                 // wypisuję okres
  for (int i=1; i<f.size(); i++)    // | wypisuję wyniki (poza okresem)
  out<<f[i]<<" ";                   // |
  out<<endl;                        // kończę wiersz
  }

//-----------------------------------------------------------------------------

void makeStarsData(string& id, string& period, vector<string>& f)
// funkcja tworzy plik z danymi na temat jednej interesującej gwiazdy
  {
  string name("output/");           // | tworzę nazwę pliku
  name+=id;                         // |
  char* ch = new char[name.size()]; // tworzę pomocniczą tablicę typu char
  for (int i=0; i<name.size(); i++) // zawierającą na miejscu i
      ch[i] = name[i];              // i-ty znak nazwy
  mkdir(ch,0777);                       // tworzę katalog o nazwie zapisanej w ch
  name+= "/data.dat" ;              // |
  for (int i=0; i<name.size(); i++) // zawierającą na miejscu i
      ch[i] = name[i];              // i-ty znak nazwy
  ofstream out(ch);                 // tworzę plik o odpowiedniej nazwie
  out<<period<<" ";                 // wypisuję okres
  for (int i=1; i<f.size(); i++)    // | wypisuję wyniki (poza okresem)
      out<<f[i]<<" ";               // |
  out.close();                      // zamykam plik
  delete[] ch;                      // usuwam pomocniczą tablicę z nazwą
  }

//-----------------------------------------------------------------------------

void makeStarsCommands(string& id, ofstream& GNUplot)
// funkcja tworzy komendy rysujące wykresy wyszczególniające gwiazdę o danym id
  {
  for (int i=0;i<6;i++)
      {
      makeCommandsBaseR(GNUplot,i,id);
      GNUplot<<", \""<<id<<"/data.dat\" using (log($1)):"
             <<(2*i+20)<<":"<<(2*i+21)<<" with yer title \"id="<<id<<"\"; \n";
      }
  makeCommandsBaseRT(GNUplot,id);
  GNUplot<<", \""<<id<<"/data.dat\" using (log($1)):32 with points title \"id="<<id<<"\" ; \n";
  }
```

```
//---------------------------------------------------------------------------

 void makeData(ofstream& GNUplot)
// funkcja tworzy pliki z danymi dla programu GNUplot
  {
  ifstream fourier("roboczy/FourierOut.txt");  // otwieram plik wynikowy z Cepheusa
  ifstream types;                              // deklaracja pliku z typami
  ofstream DCEP_FO("output/DCEP-FO.dat"),      // / pliki wynikowe
           DCEP_FU("output/DCEP-FU.dat"),      // |
           CW_FO("output/CW-FO.dat");          // |
  vector<string> f,v;                          // / zmienne
  string s,id,jd, type, period, trash;         // |
  ofstream out;                                // |
  f.clear();                                   // na wszelki wypadek czyszczę wektor
  while( fourier>>s )                          // czytam plik wynikowy z cepheusa
     {
     if ( s[s.size()-1] == 't' )               // każdy badany plik miał
                                               // rozszerzenie .dat; jeśli wczytany
                                               // wyraz jest ostatnim w lini,to:
        {
        id.clear();                            // czyszczę id
        type.clear();                          // czyszczę typ
        for (int i=0; s[i]!='_' && s[i]!='.' ; i++)
            id.push_back(s[i]);                // zapisuję nowe id
        types.open("roboczy/Types.txt");       // types - plik z informacjami o typie
        while (type.size() == 0)               // dopóki nie poznam typu:
              {
              types>>jd;                       // czytam pierwszą kolumnę
              if (jd == id)                    // jeśli to właściwy wiersz...
                 {
                 types>>period;                //...to czytam okres z drugiej kolumny
                 types>>type;                  // ... i typ z trzeciej
                 }
              else                             //jeśli to nie jest właściwy wiersz...
                 for (int i=0; i<2; i++)
                     types>>trash;             // ... to pomijam następne kolumny
              }
        types.close();                         // zamykam plik z typami
        if (type == "DCEP-FO")                 // |wybieram,w którym pliku będę pisał
           tuPisz(DCEP_FO,period,type,f);      // |
        else if (type == "DCEP-FU")            // |
           tuPisz(DCEP_FU,period,type,f);      // |
        else if (type == "CW-FO")              // |
           tuPisz(CW_FO,period,type,f);        // |
        else                                   // |
           {                                   // |
           makeStarsData(id,period,f);         // |
           makeStarsCommands(id,GNUplot);      // |
           }                                   // |
        f.clear();                             // czyszczę wektor
        }
     else              // jeśli nie jest ostatni, to jest parametrem dopasowania
        {
        f.push_back(s); // czytam do wektora współczynniki dopasowania
        }
     }
  fourier.close();      // | zamykam pliki
  DCEP_FO.close();      // |
  DCEP_FU.close();      // |
  CW_FO.close();        // |
  }

//---------------------------------------------------------------------------
```

```
int main()
// funkcja główna programu
  {
  cout<<"pracuję..."<<endl;   // komunikat dla użytkownika
  mkdir("output",0777);       // tworzę folder output
  ofstream GNUplot("output/commands.pg");
  makeData(GNUplot);          // tworzę pliki z danymi
  GNUplot.close();
  system(" cd /media/Data/Dokumenty/Pi/output; gnuplot -persist commands.pg ");
                        // używam programu GNUplot do zrobienia wykresów
  cout<<"skończyłem!"<<endl;  // komunikat dla użytkownika
  return 0;
  }
```