Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Darwin

autor	Łukasz Ochwat
prowadzący	mgr inż. Marek Kokot
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium / ćwiczeń	poniedziałek, 10:15 – 11:45
grupa	1
sekcja	2
termin oddania sprawozdania	2019-01-07
data oddania sprawozdania	2019-01-07

1 Treść zadania

Napisac program symulujacy ewolucje populacji osobników. Populacja moze liczyc dowolna liczbe osobników.
Kazdy osobnik zawiera chromosom, który jest ciagiem liczb naturalnych. Chromosomy moga byc róznej długosci. W kazdym pokoleniu wylosowywanych jest k par osobników, które sie nastepnie krzyzuja. Krzyzowanie polega na tym, ze u kazdego osobnika dochodzi do pekniecia chromosomu w dowolnym miejscu. Czesc poczatkowa chromosomu jednego osobnika łaczy sie z czescia koncowa drugiego. Inaczej mówiac: osobniki wymieniaja sie fragmentami swoich chromosomów. Jeden osobnik moze byc wylosowany do kilku krzyzowan. Po dokonaniu wszystkich krzyzowan w pokoleniu sprawdzane jest przystosowanie osobników do warunków srodowiska. W tym celu dla kazdego osobnika wyznaczana jest wartosc f 2 [0, 1] funkcji dopasowania. Osobniki, dla których wartosc f ¡ w (gdzie w jest progiem wymierania), sa usuwane z populacji. Osobniki, dla których f ¿ r (gdzie r jest progiem rozmnazania) sa klonowane. A osobniki, dla których w 6 f 6 r pozostaja w populacji, ale sie nie rozmnazaja. Program uruchamiany jest z linii polecen z wykorzystaniem nastepujacych przełaczników (kolejnosc przełaczników jest dowolna):
-i plik wejsciowy z populacja
-o plik wyjsciowy z populacja
-w współczynnik wymierania w 2 [0, 1]
-r współczynnik rozmnazania r 2 [0, 1]
-p liczba pokolen p
-k liczba k par osobników losowanych do krzyzowania
Plik wejsciowy ma nastepujaca postac: Kazda linia zawiera jednego osobnika. Osobnik charakteryzowany jest chromosomem, który jest przedstawiony jako ciag liczb naturalnych rozdzielonych białymi znakami. Przykładowy plik wejsciowy zawierajacy populacje złozona z czterech osobników: 2 9 84 9 5 6 25 12
2 98 56 2 54
5 2
8 5 22 5 48 6 1 9 8 7 554 25 235 32
Plik wyjsciowy ma identyczny format.

2 Analiza zadania

Program przedstawia problem dzielenia list na części oraz łączenia dwóch list w jedną. Z pliku wejściowego o strukturze

```
2 9 84 9 5 6 25 12
2 98 56 2 54
5 2
8 5 22 5 48 6 1 9 8 7 554 25 235 32
...
```

należy wyprowadzic plik o takiej samej strukturze, w którym będą wypisane elementy nowych list, stworzonych poprzez połączenie podzielonych w losowym miejscu list z pliku wejściowego.

2.1 Struktury danych

W programie wykorzystano jedną strukturę danych - listę dwukierunkową. Listy tej użyłem do przechowywania liczb składających się na chromosom. Dzięki użyciu typów generycznych bardzo łatwo mogłem również stworzyć listę list, która posłużyła mi do przechowywania już całych chromosomów.

2.2 Algorytmy

Program pobiera z pliku wejściowego wszystkie liczby i wczytuje je do listy chromosome znajdującej się w obiekcie klasy Subject, po czym dodaje ten obiekt do listy wszystkich osobników - subjects. Później losowane jest k par osobników, które zostają poddane krzyżowaniu, lista Chromosom każdego osobnika z pary jest dzielona na dwie części, po czym początkowa część pierwszego osobnika jest łaczona z końcową częścią drugiego. Po krzyżowaniu na podstawie odpowiedniego algorytmu* obliczane jest jego przystosowanie do przeżycia i jeśli jest większe niż parametr r to osobnik jest kopiowany, jeśli mniejsza niż parametr w to zostaje on usunięty z listy subjects, a wylosowana wartość jest mniejsza r i większa od w, to nic się nie dzieje. *algorytm oblicza przystosowanie osobnika dzieląc sumę elementów jego chromosomu przez największa sumę elementów chromosomu występującą w populacji.

3 Specyfikacja zewnętrzna

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna): -i plik wejsciowy z populacja
-o plik wyjsciowy z populacja
-w współczynnik wymierania w 2 [0, 1]
-r współczynnik rozmnazania r 2 [0, 1]

-p liczba pokolen p
-k liczba k par osobników losowanych do krzyzowania

W razie podania błędnych danych, wywoływana jest funkcja:

```
1 void ArgumentParser::showHelp()
2 {
3     std::cerr << " Something went wrong, we'll try to help you : )" << std::endl;
4     std::cerr << "Every option should have just one argument." << std::endl;
5     std::cerr << "You have to give all options:" << std::endl;
6     std::cerr << "-k amount of crossing pairs, should be a number" << std::endl;
7     std::cerr << "-r multiplication factor, should be a number between 0 and 1, greater than -w" << std::endl;
8     std::cerr << "-w exctinction factor should be a number between 0 and 1, smaller than -r" << std::endl;
9     std::cerr << "-i input file" << std::endl;
10    std::cerr << "-o output file" << std::endl;
11    std::cerr << "-p number of generations, should be a number" << std::endl;
12
13 }
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (przetwarzania chromosomów).

4.1 Typy zdefiniowane w programie

W programie zdefiniowano następujące klasy:

```
1
2 template<class Type>
3 class ListIterator
```

```
4 {
5
6 private:
7     List<Type>* list;
8     Element<Type>* pHead;
9     Element<Type>* current;
10    Element<Type>* pTail;
11 public:
12    ListIterator(List<Type>& _list);
13    void reset();
14    void reset(Element<Type>*& _current);
15    Element<Type>* begin();
16    Element<Type>* getCurrent();
17    Element<Type>* end();
18    ListIterator<Type>& operator++();
19    ListIterator<Type>& operator--();
20
21 };
```

Iterator pozwalający poruszać się po liscie

```
1 class ArgumentParser
2 {
3 private:
4     CommandLineArguments commandLineArguments;
5     bool isCorrect;
6     void parseArguments(int argumentsAmount, char*
          arguments[]);
7 public:
8     bool getIsCorrect();
9     ArgumentParser(int argumentsAmount, char* arguments
          []);
10    CommandLineArguments getCommandLineArguments();
11    void showArguments();
12    void showHelp();
13 };
```

Parser, służący do poprawnego odczytania argumentów z pliku.

```
1
2 struct CommandLineArguments
3 {
```

```
4     std::string inputFile="";
5     std::string outputFile="";
6     double extinctionFactor=-1;
7     double multiplicationFactor=-1;
8     int generationsNumber=-1;
9     int crossingPairsAmount=-1;
10 };
```

Struktura, przechowująca wszystkie argumenty wczytane z linii poleceń.

```
1   class EvolutionSimulator
2 {
3 private:
4     ArgumentParser* argParser;
5     List<Subject> subjects;
6     List<Subject> children;
7     int strongestSubjectChromosomeSum;
8 public:
9     EvolutionSimulator(ArgumentParser*& _argParser);
10    void checkSubjectsAdaptation(List<Subject>&
          _subjects);
11    List<Subject> getChildren();
12    void addSubjectsToPopulation(List<Subject>&
          _subjects);
13    bool loadSubjects();
14    bool saveSubjectsToFile();
15    void simulateEvolution();
16 };
```

główna klasa programu zajmująca się wczytaniem i zapisaniem danych do pliku, oraz symulująca ewolucję w sposób opisany w sekcji Algorytmy.

```
1   template <class Type>
2 struct Element
3 {
4     Type value;
5     Element * pNext;
6     Element * pPrev;
7 };
8
9 template <class Type>
10 class List
11 {
```

```
12 private:
13     Element<Type>* pHead;
14     Element<Type>* pTail;
15     unsigned int size;
16 public:
17     /**
18     sets size to 0, pHead and pTail to nullptr
19     */
20     List<Type>() {
21     size = 0;
22     pHead = nullptr;
23     pTail = nullptr;
24     }
25     List<Type>(const List<Type>& other);
26
27     /**
28     deletes list iteratively
29     */
30     ~List<Type>() {
31         deleteIteratively();
32     }
33     Element<Type>* head();
34     Element<Type>* tail();
35     void addAtBeginning(Type value);
36     void addAtEnd(Type value);
37     void deleteElement(Element<Type>* pElement);
38     void writeIteratively();
39     void writeFromTheEndIteratively();
40     void deleteIteratively();
41     unsigned int const getSize();
42     List<Type>& operator=(const List<Type>& other);
43 };
```

Główna klasa listy, pozwalająca na dodawanie obiektów na jej początek i koniec, usuwanie całej listy (oraz poszczególnych obiektów), wypisywanie jej od początku i od końca.

```
1  class Subject
2 {
3 private:
4     List<unsigned int> chromosome;
5     double f;
```

```
6 public:
7     Subject(const List<unsigned int>& _chromosome);
8     Subject(List<unsigned int>& firstChromosomePart,
          List<unsigned int>& secondChromosomePart);
9     List<unsigned int> getChromosome();
10    int getSumOfChromosomeNumbers();
11    std::pair<List<unsigned int>, List<unsigned int>>
          split();
12    double getAdaptation(int strongestSum);
13 };
```

klasa reprezentująca osobnika, zawiera jego chromosom w postaci listy oraz przystosowanie do środowiska, pozwala uzyskać parę list utworzonych po podzieleniu listy na dwie części

4.2 Ogólna struktura programu

W funkcji main tworzony jest ArgumentParser, który przyjmuje jako argumenty argc i argv[], czyli argumenty maina.

```
1 ArgumentParser* argParser = new ArgumentParser(argc,
      argv);
```

Później, wewnątrz ArgumentParsera parsowane są argumenty podane przez użytkownika i w razie błednych danych flaga parsera isCorrect ustawiana jest na false. Stan tej flagi jest później sprawdzany i w zależności od niej, program wykona się bądź też wyświetli komunikat o błędzie.

```
1 if (argParser->getIsCorrect())
2     {
3         EvolutionSimulator evolutionSimulator(argParser
              );
4         evolutionSimulator.simulateEvolution();
5     }
```

Jeśli flaga jest ustawiona na true, tworzymy obiekt klasy EvolutionSimulator i jako argument podajemy stworzony argParser. Potem wywołujemy metode klasy EvolutionSimulator - simulateEvolution(), która odpowiada za obsługę całego algorytmu.

W metodzie simulateEvolution próbujemy wczytać wszystkie chromosomy z pliku. Jeśli się to uda - przechodzimy dalej, w przeciwnym razie program wraca do funckji main, gdzie czyszczona jest pamięć i program się kończy.

```
1 if (!loadSubjects())
2   {
3     return;
4   }
```

Później, w pętli, wykonującej się tyle razy, ile wynosi podany przez użytkownika parametr p, do listy osobników children, reprezentującą osobniki pretendujące do bycia dodanymi do populacji, przypisywana jest lista zwrócona przez metodę getChildren(). Metoda ta zwraca listę, składającą się z osobników utworzonych w sposób opisany w sekcji Algorytmy. Później sprawdzana jest adaptacja osobników z listy children w metodzie checkSubjectsAdaptation(), na końcu wszystkie pozostałe osobniki zostają dodane do populacji przy użyciu funkcji addSubjectsToPopulation()

```
1     for (int i = 0; i < argParser->
          getCommandLineArguments().generationsNumber; i
          ++)
2   {
3     children = getChildren();
4     checkSubjectsAdaptation(children);
5     addSubjectsToPopulation(children);
6   }
```

Po wykonaniu się algorytmów, wszystkie osobniki z listy subjects zostają zapisane do pliku przez funkcję

```
1     saveSubjectsToFile();
```

Po wszystkim, w mainie zwalniana jest pamięć.

```
1       delete argParser;
2     std::cin.ignore();
3      return 0;
```

4.3 Szczegółowy opis implementacji funkcji

Szczegółowy opis typów i funkcji został dołączony w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach - podanie pliku, w którym zamiast liczb występują litery kończy się pojawieniem się błędu.

W razie niepoprawnego podania argumentów w linii poleceń lub podania ich zbyt małej / zbyt dużej ilości również kończy się błędem.

6 Wnioski

Idea programu byla prosta do zrozumienia, co od razu pozwolilo mi przejść do dzialania. Samo zajęcie się algorytmem nie sprawiło mi większego problemu, chociaż podczas pracy nad nim pojawiło się parę małych błędów, które znacząco zmieniały działanie programu, a które cieżko było wykryć. Najwięcej czasu zajęła mi implementacja klas w przejrzysty sposób i zajęcie się takimi niuansami jak konstruktory kopiujące, operatory przypisania czy typy generyczne. Mimo tego, że implementacja tego zajęła mi trochę czasu, odczułem, że było to opłacalne posunięcie. Dzięki niemu np. stworzenie listy list sprowadzało się do napisania List¡List¡unsigned int» x; - nie musiałem tworzyć osobnej struktury danych i implementowac jej od początku.

Darwin

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 ArgumentParser Class Reference

Public Member Functions

- bool getIsCorrect ()
- ArgumentParser (int argumentsAmount, char ∗arguments[ ])
- CommandLineArguments getCommandLineArguments ()
- void showArguments ()
- void showHelp ()

2.1.1 Constructor & Destructor Documentation

2.1.1.1 ArgumentParser()

```
ArgumentParser::ArgumentParser (
            int argumentsAmount,
            char * arguments[] )
```

Parameters

argumentsAmount	liczba argumentow uruchomieniowych podanych przez uzytkownika
arguments[ ]	argumenty uruchomieniowe podane przez uzytkownika wywoluje funkcje parseArguments w celu zparsowania argumentow

2.1.2 Member Function Documentation

2.1.2.1 getCommandLineArguments()

```
CommandLineArguments ArgumentParser::getCommandLineArguments ( )
```

Returns

commandLineArguments

2.1.2.2 getIsCorrect()

```
bool ArgumentParser::getIsCorrect ( )
```

Returns

isCorrect

2.1.2.3 showArguments()

```
void ArgumentParser::showArguments ( )
```

wypisuje wszystkie pola commandLineArguments na ekran

2.1.2.4 showHelp()

```
void ArgumentParser::showHelp ( )
```

shows help

The documentation for this class was generated from the following files:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/ArgumentParser.h
- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/ArgumentParser.cpp

2.2 CommandLineArguments Struct Reference

Public Attributes

- std::string **inputFile** =""
- std::string **outputFile** =""
- double **extinctionFactor** =-1
- double **multiplicationFactor** =-1
- int **generationsNumber** =-1
- int **crossingPairsAmount** =-1

The documentation for this struct was generated from the following file:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/CommandLineArguments.h

2.3 Element< Type > Struct Template Reference

Public Attributes

- Type **value**
- Element * **pNext**
- Element * **pPrev**

The documentation for this struct was generated from the following file:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/List.h

2.4 EvolutionSimulator Class Reference

Public Member Functions

- EvolutionSimulator (ArgumentParser *&_argParser)
- void checkSubjectsAdaptation (List< Subject > &_subjects)
- List< Subject > getChildren ()
- void addSubjectsToPopulation (List< Subject > &_subjects)
- bool loadSubjects ()
- bool saveSubjectsToFile ()
- void simulateEvolution ()

2.4.1 Constructor & Destructor Documentation

2.4.1.1 EvolutionSimulator()

```
EvolutionSimulator::EvolutionSimulator (
            ArgumentParser *& _argParser )
```

Parameters

_argParser	wskaznik na ArgumentParser, w ktorym przechowywane sa parametry uruchomieniowe programu

2.4.2 Member Function Documentation

2.4.2.1 addSubjectsToPopulation()

```
void EvolutionSimulator::addSubjectsToPopulation (
            List< Subject > & _subjects )
```

Parameters

lista	osobnikow, ktore maja zostac dodane do listy subjects dodaje osobniki z podanej w parametrze listy do listy subjects

2.4.2.2 checkSubjectsAdaptation()

```
void EvolutionSimulator::checkSubjectsAdaptation (
            List< Subject > & _subjects )
```

Parameters

_subjects	lista osobnikow, ktorych przystosowanie chcemy przetestowac. Oblicza przystosowanie osobnikow z podanej listy i w zaleznosci od niego usuwa, klonuje lub nic nie robi z danym osobnikiem.

2.4.2.3 getChildren()

```
List< Subject > EvolutionSimulator::getChildren ( )
```

Krzyzuje losowo wybrane pary osobnikow poprzez polaczenie poczatkowej czesci chromosomu jednego z koncowa czescia drugiego osobnika (ilosc par zalezna od parametru -k podanego przez uzytkownika).

Returns

lista osobnikow powstalych po skrzyzowaniu sie wybranych osobnikow z populacji

2.4.2.4 loadSubjects()

```
bool EvolutionSimulator::loadSubjects ( )
```

Wczytuje wszystkie osobniki z pliku podanego przez uzytkownika przy uruchamianiu programu po parametrze -i i zapisuje je do listy subjects.

Returns

true, jesli wczytanie sie powiodlo, false jestli plik nie istnieje lub w pliku sa nieprawidlowe dane

2.4.2.5 saveSubjectsToFile()

```
bool EvolutionSimulator::saveSubjectsToFile ( )
```

Zapisuje wszystkie osobniki do pliku podanego przez uzytkownika przy uruchamianiu programu po parametrze -o.

Returns

true, jesli zapisanie sie powiodlo, false jesli plik nie istnieje

2.4.2.6 simulateEvolution()

```
void EvolutionSimulator::simulateEvolution ( )
```

Glowna funkcja symulujaca ewolucje. Wczytuje osobniki z pliku. Po czym, w ptli, krzyzuje osobniki, testuje ich przystosowanie i dodaje je do populacji. Petla wykonuje sie tyle razy, ile wynosi parametr -p podany przez uzytkowika.

The documentation for this class was generated from the following files:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/EvolutionSimulator.h
- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/EvolutionSimulator.cpp

2.5 List< Type > Class Template Reference

Public Member Functions

- List ()
- **List** (const List< Type > &other)
- ~List ()
- Element< Type > * head ()
- Element< Type > * tail ()
- void addAtBeginning (Type value)
- void addAtEnd (Type value)
- void deleteElement (Element< Type > *pElement)
- void writeIteratively ()
- void writeFromTheEndIteratively ()
- void deleteIteratively ()
- unsigned int const getSize ()
- List< Type > & operator= (const List< Type > &other)

2.5.1 Constructor & Destructor Documentation

2.5.1.1 List()

```
template<class Type>
List< Type >::List ( )  [inline]
```

sets size to 0, pHead and pTail to nullptr

2.5.1.2 ~List()

```
template<class Type>
List< Type >::~List ( )  [inline]
```

deletes list iteratively

2.5.2 Member Function Documentation

2.5.2.1 addAtBeginning()

```
template<class Type>
void List< Type >::addAtBeginning (
            Type value )
```

Parameters

value	wartosc, ktora ma zostac dodana do listy dodaje element na poczatek listy

2.5.2.2 addAtEnd()

```
template<class Type>
void List< Type >::addAtEnd (
            Type value )
```

Parameters

value	wartosc, ktora ma zostac dodana do listy dodaje element na koniec listy

2.5.2.3 deleteElement()

```
template<class Type>
```

```
void List< Type >::deleteElement (
            Element< Type > * pElement )
```

Parameters

pElement	wskaznik na element, ktory ma zostac usuniety z listy usuwa podany element z listy

2.5.2.4 deleteIteratively()

```
template<class Type >
void List< Type >::deleteIteratively ( )
```

usuwa liste iteracyjnie

2.5.2.5 getSize()

```
template<class Type >
unsigned int const List< Type >::getSize ( )
```

Returns

size

2.5.2.6 head()

```
template<class Type >
Element< Type > * List< Type >::head ( )
```

Returns

pHead

2.5.2.7 operator=()

```
template<class Type>
List< Type > & List< Type >::operator= (
            const List< Type > & other )
```

przeciazony operator przypisania, usuwa liste, po czym dodaje do niej elementy z listy podanej jako prawy argument

2.5.2.8 tail()

```
template<class Type >
Element< Type > * List< Type >::tail ( )
```

Returns

pTail

2.5.2.9 writeFromTheEndIteratively()

```
template<class Type >
void List< Type >::writeFromTheEndIteratively ( )
```

wypisuje liste od konca iteracyjnie

2.5.2.10 writeIteratively()

```
template<class Type >
void List< Type >::writeIteratively ( )
```

wypisuje liste od poczatku iteracyjnie

The documentation for this class was generated from the following file:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/List.h

2.6 ListIterator< Type > Class Template Reference

Public Member Functions

- ListIterator (List< Type > &_list)
- void reset ()
- void reset (Element< Type > *&_current)
- Element< Type > * begin ()
- Element< Type > * getCurrent ()
- Element< Type > * end ()
- ListIterator< Type > & operator++ ()
- ListIterator< Type > & operator-- ()

2.6.1 Constructor & Destructor Documentation

2.6.1.1 ListIterator()

```
template<class Type >
ListIterator< Type >::ListIterator (
            List< Type > & _list )
```

Parameters

_list	lista, ktorej obiekt bedzie iteratorem. przypisuje podana liste do pola list, glowe listy do pola pHead, ogon listy do pola pTail i ustawia pole current na pHead.

2.6.2 Member Function Documentation

2.6.2.1 begin()

```
template<class Type >
Element< Type > * ListIterator< Type >::begin ( )
```

Returns

pHead

2.6.2.2 end()

```
template<class Type >
Element< Type > * ListIterator< Type >::end ( )
```

Returns

pTail

2.6.2.3 getCurrent()

```
template<class Type >
Element< Type > * ListIterator< Type >::getCurrent ( )
```

Returns

current

2.6.2.4 operator++()

```
template<class Type >
ListIterator< Type > & ListIterator< Type >::operator++ ( )
```

przeciazenie operatora inkrementacji, ustawia current na current->pNext

2.6.2.5 operator--()

```
template<class Type >
ListIterator< Type > & ListIterator< Type >::operator-- ( )
```

przeciazenie operatora dekrementacji, ustawia current na current->pPrev

2.6.2.6 reset() [1/2]

```
template<class Type >
void ListIterator< Type >::reset ( )
```

resetuje iterator, poprzez ponowne ustawienie pHead i pTail na, kolejno, glowe i ogon listy, oraz pola current na pHead.

2.6.2.7 reset() [2/2]

```
template<class Type >
void ListIterator< Type >::reset (
            Element< Type > *& _current )
```

Parameters

_current	wskaznik na pozadany element, na ktory ma wskazywac iterator resetuje iterator, poprzez ponowne ustawienie pHead i pTail na, kolejno, glowe i ogon listy, ale ustawia pole current na podane przez uzytkownika w parametrze.

The documentation for this class was generated from the following file:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/ListIterator.h

2.7 Subject Class Reference

Public Member Functions

- Subject (const List< unsigned int > &_chromosome)
- Subject (List< unsigned int > &firstChromosomePart, List< unsigned int > &secondChromosomePart)
- List< unsigned int > getChromosome ()
- int getSumOfChromosomeNumbers ()
- std::pair< List< unsigned int >, List< unsigned int > > split ()
- double getAdaptation (int strongestSum)

2.7.1 Constructor & Destructor Documentation

2.7.1.1 Subject() [1/2]

```
Subject::Subject (
            const List< unsigned int > & _chromosome )
```

Parameters

_chromosome	lista intow reprezentujaca chromosom osobnika przypisuje _chromosome do pola chromosome

2.7.1.2 Subject() [2/2]

```
Subject::Subject (
            List< unsigned int > & firstChromosomePart,
            List< unsigned int > & secondChromosomePart )
```

Parameters

firstChromosomePart	lista intow reprezentujaca pierwsza czesc chromosomu osobnika
secondChromosomePart	lista intow reprezentujaca druga czesc chromosomu osobnika tworzy osobnika, ktorego pole chromosome to polaczone listy firstChromosomePart i secondChromosomePart

2.7.2 Member Function Documentation

2.7.2.1 getAdaptation()

```
double Subject::getAdaptation (
            int strongestSum )
```

Parameters

strongestSum	najwieksza suma elementow chromosomu w calej populacji oblicza przystosowania osobnika dzielac sume elementow jego chromosomu przez strongestSum

Returns

przystosowanie osobnika

2.7.2.2 getChromosome()

```
List< unsigned int > Subject::getChromosome ( )
```

Returns

chromosome

2.7.2.3 getSumOfChromosomeNumbers()

```
int Subject::getSumOfChromosomeNumbers ( )
```

sumuje wszystkie elementy listy chromosome

Returns

suma wszystkich elementow listy reprezentujacej chromosom (pole chromosome)

2.7.2.4 split()

```
std::pair< List< unsigned int >, List< unsigned int > > Subject::split ( )
```

dzieli liste chromosome w losowym miejscu, tworzac w ten sposob dwie nowe listy

Returns

para list unsigned int, reprezentujaca dwie czesci chromosomu

The documentation for this class was generated from the following files:

- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/Subject.h
- C:/Users/Mumin/source/repos/48ee7d50-gr02-repo/projekt/Darwin/Darwin/Subject.cpp

Index