Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Temat 4.: Darwin

autor	Piotr Głąb
prowadzący	mgr inż. Maciej Długosz
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
1. termin laboratorium	wtorek, 09:00 – 12:15
2. termin laboratorium	środa, 08:15 – 10:30
sekcja	21
termin oddania sprawozdania	2019-11-13

1 Treść zadania

Napisać program symulujący ewolucję populacji osobników. Populacja może liczyć dowolną liczbę osobników. Każdy osobnik zawiera chromosom, który jest ciągiem liczb naturalnych. Chromosomy mogą być różnej długości. W każdym pokoleniu wylosowywanych jest k par osobników, które się następnie krzyżują. Krzyżowanie polega na tym, że u każdego osobnika dochodzi do pęknięcia chromosomu w dowolnym miejscu. Cześć początkowa chromosomu jednego osobnika łączy się z częścią końcową drugiego. Inaczej mówiąc: osobniki wymieniają się fragmentami swoich chromosomów. Jeden osobnik może być wylosowany do kilku krzyżowań. Po dokonaniu wszystkich krzyżowań w pokoleniu sprawdzane jest przystosowanie osobników do warunków środowiska. W tym celu dla każdego osobnika wyznaczana jest wartość $f \in [0, 1]$ funkcji dopasowania. Osobniki, dla których wartość $f < w$ (gdzie w jest progiem wymierania), są usuwane z populacji. Osobniki, dla których $f > r$ (gdzie r jest progiem rozmnażania) są klonowane. A osobniki, dla których $w \leq f \leq r$ pozostają w populacji, ale się nie rozmnażają. Sposób określania wartości f w oparciu o postać osobnika jest zależna od problemu. Można przyjąć, że jest to średnia arytmetyczna wartości wchodzących w skład chromosomu osobnika. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):

- `-i` plik wejściowy z populacją
- `-o` plik wyjściowy z populacją
- `-w` współczynnik wymierania $w \in [0, 1]$
- `-r` współczynnik rozmnażania $r \in [0, 1]$
- `-p` liczba pokoleń p
- `-k` liczba k par osobników losowanych do krzyżowania

2 Analiza zadania

Zagadnienie przedstawia problem rozwoju populacji osobników reprezentowanych poprzez ich chromosomy będące ciągami liczbowymi umieszczonymi w pliku wejściowym w osobnych liniach. Realizacja procesu ewolucji polega na krzyżowaniu się genów osobników danego pokolenia w sposób losowy podobnie jak w przypadku rozmnażania się prawdziwej populacji. Następnie w wyniku konfrontacji z warunkami środowiska (odpowiednia funkcja dopasowania) osobniki słabsze wymierają, zaś silniejsze tworzą nowe pokolenie (z przewagą występowania osobników najsilniejszych - stąd klonowanie).

2.1 Struktury danych

W programie wykorzystano kontenery STL, a dokładnie mówiąc listy. Narzucenie przez prowadzącego realizacji programu przy pomocy tych struktur wydaje mi się być spowodowane przede wszystkim dwiema rzeczami. Konkretnie rzecz biorąc, listy pozwalają na bardzo szybkie dodawanie i usuwanie elementów, a adresy elementów są niezmienne, co odróżnia je od na przykład wektora.

Takie rozwiązanie ułatwia znacząco kontrolowanie wylosowanych do krzyżowań chromosomów czy miejsc przerwania ich (w celu dokonania tej operacji) i przyspiesza działanie programu wobec konieczności dokonywania w populacji częstych zmian.

2.2 Algorytmy

Logika programu jest realizowana w dwóch zasadniczych algorytmach: krzyżującym i konfrontującym pokolenie ze środowiskiem.

Po pierwsze: program w pętli iteracyjnej kontrolującej ilość dokonanych krzyżowań losuje chromosomy do krzyżowania przy pomocy ilości elementów w populacji, a następnie tworzy kopie lokalne chromosomów wylosowanych, aby uniknąć zbyt wielu iteracji przez całą populację. Na kopiach dokonuje odpowiednich zmian by na końcu wstawić (w stałym czasie) zmodyfikowane kopie chromosomów w miejsce pobranych z oryginalnej populacji. Wykonanie tej pętli jest równoznaczne z zakończeniem etapu krzyżowania.

Po drugie: program testuje dopasowanie osobników w populacji po dokonaniu krzyżowań. W pętli iterującej po populacji wylicza wartość funkcji dopasowania każdego chromosomu i w zależności od uzyskanej wartości przyporządkowuje osobnika do nowej listy tworząc tym samym nowe pokolenie, które jest wstawiane później w miejsce starego.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: `-i` dla pliku wejściowego i `-o` dla pliku wyjściowego), a także współczynniki wymierania `-w` i rozmnażania `-r` będące liczbami zmiennoprzecinkowymi z zakresu [0,1] oraz liczby pokoleń `-p` i par do krzyżowania `-k` będące liczbami całkowitymi nieujemnymi. Poniżej prezentuję dwa przykłady:

```
nazwa_programu -i plik_we.txt -o plik_wy -w 0.3 -r 0.75 -p 2 -k 5
```

```
nazwa_programu -o plik_wy.txt -r 0.8 -p 1 -w 0.4 -k 0 -i plik_we.txt
```

Jak można zauważyć na przykładzie przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez żadnego parametru, z nieprawidłowymi parametrami (złą ich ilością lub błędnym formatem wśród tych liczbowych) lub z parametrem `-h`

```
program
program -h
```

powoduje wyświetlenie komunikatu o nieprawidłowych parametrach i instrukcji obsługi. Pliki są plikami tekstowymi, ale mogą mieć dowolne rozszerzenie lub go nie mieć (więcej na temat rozszerzeń w sekcji 5 Testowanie). Plik wejściowy musi jednak zawierać odpowiednie dane (właściwie nie musi, przy czym dla pustego pliku wejściowego otrzymamy pusty plik wyjściowy). Konkretnie rzecz biorąc powinien zawierać liczby całkowite oddzielone białymi znakami umieszczone w liniach (ciąg liczbowy w jednej linii odpowiadać będzie jednemu osobnikowi). Dla przykładu:

```
2 4 239 47 2 4 5
1 34 3 2
1 4
0
2 4 5 2
```

Plik wyjściowy przyjmuje taki sam format. W razie nie otwarcia któregokolwiek z plików (np. w skutek podania nieprawidłowej nazwy lub rozszerzenia) program wyświetla odpowiedni komunikat z spośród dwóch przedstawionych poniżej:

```
Nie otwarto pliku wejściowego.
Nie otwarto pliku wyjściowego
```

a następnie pomoc. Więcej informacji na temat interpretowania przez program błędów związanych z plikami w sekcji 5 Testowanie. Przykładowe pliki dołączono do programu.

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (tworzenia populacji i jej modyfikowania). Wspomniana komunikacja

odbywa się zgodnie z założeniem zadania o posługiwaniu się przełącznikami: funkcje sprawdzające poprawność przełączników, zwracające komunikaty o błędach i funkcja wyświetlająca pomoc są umieszczone w pliku źródłowym widok.cpp i odpowiadającym mu pliku nagłówkowym widok.h. Dla odróżnienia: operacje na plikach (wczytanie, zapisanie) czy wszelkie modyfikacje populacji lub testy w poszukiwaniu błędów (z zastrzeżeniem o byciu wywoływanymi przez funkcję widoku) znajdują się w pliku źródłowym model.ccp oraz odpowiadającym mu pliku nagłówkowym model.h. Wszystkie funkcje zagnieżdżone wywoływane są pośrednio przez otaczającą je funkcję umieszczoną w funkcji main w osobnym pliku źródłowym main.cpp

4.1 Ogólna struktura programu

W funkcji głównej w pętli while wywoływana jest alternatywa funkcji: ilosc_przelacznikow wywołanej jednokrotnie oraz funkcji pobierz_przelacznik wywołanej dla każdego przełącznika, jaki powinien podać użytkownik. Jeśli którekolwiek z powyższych wywołań zwróci wartość **false** program wywołuje funkcję instrukcja_obslugi i kończy swoje działanie.

W wypadku poprawnego podania przełączników następuje kolejny test: jeżeli funkcja sprawdz_poprawnosc_przelacznikow sprawdzająca, czy przełączniki k,p,w,r przyjmują odpowiedni format liczbowy, zwraca wartość **true** następuje konwersja wyżej wymienionych argumentów z łańcucha znakowego na odpowiedni typ liczbowy. W przeciwnym wypadku wyświetlana jest instrukcja obsługi i program kończy swoje działanie. Następnie tworzone są odpowiednie strumienie plikowe, których otwarcie sprawdzane jest funkcją sprawdz_otwarcie i znów wyświetlane są odpowiednie komunikaty oraz instrukcja obsługi w razie wystąpienia błędów.

W tym momencie może wreszcie nastąpić pobranie danych do populacji funkcją wczytaj_z_pliku , wywoływaną w instrukcji warunkowej przyzywającej dodatkowo funkcję litera_w_pliku w razie wystąpienia podobnego błędu we wczytaniu danych.

Oto dochodzimy do zasadniczej części programu: w instrukcji warunkowej sprawdzającej czy populacja zawiera jakiekolwiek osobniki wywoływane są funkcje modyfikuj_przez_pokolenia oraz wypisz_do_pliku jeśli populacja jest zapełniona danymi. Jeśli na tym etapie nie była, znaczy to, iż plik wejściowy był pusty, co zwalnia z konieczności jego modyfikacji i pozwala przejść do zakończenia działania programu. Jeśli w całym procesie nie wystąpił żaden błąd, program nie wyświetli żadnej instrukcji na ekran i wyłączy się.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki niepoprawne (niezawierające liczb, zawierające liczby w niepoprawnym formacie, niezgodne ze specyfikacją) powodują zgłoszenie błędu. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego (został podany pusty zbiór liczb i pusty zbiór został zwrócony). Chromosomy w pliku wejściowym zawierające wyłącznie jeden gen są poddawane krzyżowaniu poprzez oddanie całego chromosomu do drugiego krzyżowanego, a w miejsce początkowe wpisaniu jedynie fragmentu tegoż. Jeśli chodzi o wielkość plików, to największą bazą liczb, jakiej użyto do testów był plik zawierający milion osobników o chromosomach długości do stu genów, będących liczbami z przedziału od 0 do 100. Użyty plik wejściowy miał około 36kB. Nie wywołał błędu alokacji pamięci. Zrezygnowano jednak z dalszych testów z powodu dojścia do wniosku, że już dla takich wartości program jest stosunkowo niewydajny - wykonywał się około dwóch i pół minuty (mierzone z dokładnością do jednej sekundy) - toteż stosowanie go dla większych zbiorów będzie niepraktyczne. Dla przeciętnego zbioru o ilości do dziesięciu tysięcy osobników (o podobnych parametrach jak w większym pliku) program zwraca wyniki w czasie poniżej 4 sekund (mierzone z dokładnością do jednej sekundy)

Program został przetestowany pod kątem wycieków pamięci przy pomocy profilera wydajności programu Microsoft Visual Studio.

Nawiązując do odwołań zawartych w sekcji 3 Specyfikacja zewnętrzna po pierwsze: można następująco opisać zagadnienie rozszerzeń w plikach wejściowych i wyjściowych.

Program testowano z plikami wejściowymi i wyjściowymi w formatach:

```
.txt .doc .docx .rtf "bez rozszerzenia"
```

Program działa z każdym z wymienionych rozszerzeń oprócz .docx (oczywiście w wypadku uwzględnienia zmiany rozszerzenia w chwili podawania przełącznika do programu). Istnieje możliwość podania równocześnie różnych formatów pliku wejściowego oraz wyjściowego. Co do pliku bez rozszerzenia można go stworzyć przykładowo w następujący sposób: po zapisaniu danych wejściowych w pliku .txt zwyczajnie usunąć rozszerzenie z pliku zmieniając

nazwę. Podobnie z plikiem wyjściowym. Ciężko wskazać zastosowania dla takiego rozwiązania, ale dla samej idei testowania różnych możliwości (nawet skrajnych) sprawdzono również tą. Program zachował się normalnie (oczywiście dla pliku wyjściowego należało wymusić inne rozszerzenie, by móc odczytać wynik). Takie działanie przywiodło myśl przetestowania ręcznych zmian rozszerzeń - dane w pliku wyjściowym lub wejściowym zachowują swój format i treść przy wymuszaniu zmian rozszerzeń pomiędzy wszystkimi wyżej wspomnianymi (otwieranie plików testowano za pomocą programów: notatnik oraz MO Word 2007). Można zewnętrznie zmienić rozszerzenie i znów użyć plików w programie.

Po drugie: można zgłębić zagadnienie interpretacji błędów w pliku wejściowym. Zasadniczo program interpretuje znaki podane w pliku wejściowym i wymusza na użytkowniku podanie danych zgodnych z wymogami zadania. Po wykryciu w pliku wejściowym innego znaku niż liczba całkowita program przerywa działanie i wyświetla stosowny komunikat nakazujący poprawę. Początkowo istniała koncepcja interpretowania znaków innych niż liczby jako zera. Zrezygnowano jednak z niej z racji, że takie wartości (mylnie) zaniżałyby wartość funkcji dopasowania chromosomów danego osobnika. Ponadto program postrzega brak znaku końca linii w ostatnim wierszu jako błąd, ale sam go dopisuje, by nie zmuszać użytkownika do interpretowania tak trywialnych zagadnień, gdyż w gruncie rzeczy nie jest to błąd w świetle zadanego formatu pliku wejściowego, a jedynie element uniemożliwiający programowi pracę ze względu na używanie metody `.eof()` z biblioteki <sstream> we wczytywaniu z pliku.

6 Wnioski

Program analizujący rozwój populacji osobników jest programem interesującym, na swój sposób złożonym i wymagającym odpowiedniego zarządzania pamięcią, choć po przeanalizowaniu treści nie jest zbyt skomplikowany. Najbardziej wymagające okazało się interpretowanie danych wejściowych w pliku oraz przełączników w taki sposób, aby uniknąć ewentualnych błędów. Dopracowanie programu pod kątem ochrony przed choćby nieopatrznymi błędami było bardzo ciekawym zagadnieniem, na którego analizę poświęciłem sporą ilość czasu. Również przemyślenie algorytmów działających na populacji w taki sposób, by zużywać możliwie mało pamięci i nie iterować się ciągle po populacji (ostatecznie wykorzystano kopie osobników krzyżowanych, na których dokonuje się zmian i które potem wpisuje się do populacji) okazało się jednym z ważniejszych elementów dla optymalizacji programu. Program

okazał się w miarę wydajny - dla populacji dziesięciu tysięcy osobników czas jego pracy nie przekracza 3-4 sekund. Bardzo interesującym zagadnieniem okazała się także interpretacja rozszerzeń dla plików wejściowych oraz wyjściowych. Cały program w prawidłowy dla treści zadania sposób interpretuje podane mu wartości.

Dodatek
Szczegółowy opis typów i funkcji

Projekt Darwin

Autor: Piotr Głąb

Wygenerowano przez Doxygen 1.8.20

1 README 1

2 Indeks plików 3

2.1 Lista plików . . . 3

3 Dokumentacja plików 5

3.1 Dokumentacja pliku Darwin.vcxproj.FileListAbsolute.txt . . . 5

3.2 Dokumentacja pliku Main.cpp . . . 5

3.2.1 Dokumentacja funkcji . . . 5

3.2.1.1 main() . . . 5

3.3 Dokumentacja pliku Model.cpp . . . 5

3.3.1 Dokumentacja funkcji . . . 7

3.3.1.1 czy_double() . . . 7

3.3.1.2 czy_int() . . . 7

3.3.1.3 dopisz_enter() . . . 8

3.3.1.4 konfrontuj_pokolenie_ze_srodowiskiem() . . . 8

3.3.1.5 konwertuj_przelaczniki() . . . 9

3.3.1.6 krzyzuj_pokolenie() . . . 9

3.3.1.7 losuj_pare() . . . 10

3.3.1.8 modyfikuj_przez_pokolenia() . . . 11

3.3.1.9 naprawa_strumienia() . . . 11

3.3.1.10 pobierz_koncowke() . . . 12

3.3.1.11 scal_chromosomy() . . . 12

3.3.1.12 sprawdz_dopasowanie() . . . 13

3.3.1.13 sprawdz_koniec() . . . 13

3.3.1.14 tworz_nowe_pokolenie() . . . 14

3.3.1.15 usun_skopiowane() . . . 14

3.3.1.16 wczytaj_z_pliku() . . . 15

3.3.1.17 wypisz_do_pliku() . . . 16

3.3.1.18 zamien_z_kopia() . . . 16

3.3.1.19 znajdz_krzyzowany() . . . 16

3.3.1.20 znajdz_najwieksza_srednia() . . . 17

3.4 Dokumentacja pliku MODEL.h . . . 17

3.4.1 Dokumentacja funkcji . . . 18

3.4.1.1 czy_double() . . . 19

3.4.1.2 czy_int() . . . 19

3.4.1.3 dopisz_enter() . . . 20

3.4.1.4 konfrontuj_pokolenie_ze_srodowiskiem() . . . 20

3.4.1.5 konwertuj_przelaczniki() . . . 21

3.4.1.6 krzyzuj_pokolenie() . . . 21

3.4.1.7 losuj_pare() . . . 22

3.4.1.8 modyfikuj_przez_pokolenia() . . . 22

3.4.1.9 naprawa_strumienia() . . . 23

3.4.1.10 pobierz_koncowke() . . . 23
3.4.1.11 scal_chromosomy() . . . 24
3.4.1.12 sprawdz_dopasowanie() . . . 25
3.4.1.13 sprawdz_koniec() . . . 25
3.4.1.14 tworz_nowe_pokolenie() . . . 26
3.4.1.15 usun_skopiowane() . . . 26
3.4.1.16 wczytaj_z_pliku() . . . 27
3.4.1.17 wypisz_do_pliku() . . . 28
3.4.1.18 zamien_z_kopia() . . . 28
3.4.1.19 znajdz_krzyzowany() . . . 28
3.4.1.20 znajdz_najwieksza_srednia() . . . 29
3.5 Dokumentacja pliku plik_wejsciowy.txt . . . 29
3.6 Dokumentacja pliku plik_wyjsciowy.txt . . . 29
3.7 Dokumentacja pliku README.md . . . 29
3.8 Dokumentacja pliku Widok.cpp . . . 29
3.8.1 Dokumentacja funkcji . . . 30
3.8.1.1 ilosc_przelacznikow() . . . 30
3.8.1.2 instrukcja_obslugi() . . . 30
3.8.1.3 litera_w_pliku() . . . 31
3.8.1.4 pobierz_przelacznik() . . . 31
3.8.1.5 sprawdz_otwarcie() . . . 31
3.8.1.6 sprawdz_poprawnosc_przelacznikow() . . . 33
3.8.1.7 wypisz_populacje() . . . 33
3.9 Dokumentacja pliku WIDOK.h . . . 34
3.9.1 Dokumentacja funkcji . . . 34
3.9.1.1 ilosc_przelacznikow() . . . 34
3.9.1.2 instrukcja_obslugi() . . . 35
3.9.1.3 litera_w_pliku() . . . 35
3.9.1.4 pobierz_przelacznik() . . . 35
3.9.1.5 sprawdz_otwarcie() . . . 36
3.9.1.6 sprawdz_poprawnosc_przelacznikow() . . . 36
3.9.1.7 wypisz_populacje() . . . 37

Indeks **39**

Rozdział 1

README

Tutaj umieszczaÄ‡ projekt

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Main.cpp 5
Model.cpp 5
MODEL.h 17
Widok.cpp 29
WIDOK.h 34

Rozdział 3

Dokumentacja plików

3.1 Dokumentacja pliku Darwin.vcxproj.FileListAbsolute.txt

3.2 Dokumentacja pliku Main.cpp

```
#include "WIDOK.h"
#include "MODEL.h"
```

Funkcje

- int main (int args, char ∗params[ ])

3.2.1 Dokumentacja funkcji

3.2.1.1 main()

```
int main (
            int args,
            char * params[] )
```

3.3 Dokumentacja pliku Model.cpp

```
#include "MODEL.h"
```

Funkcje

- bool czy_int (const string &przelacznik_calkowitoliczbowy)

  Funkcja sprawdzająca, czy podany ciąg znaków zawiera liczbę całkowitą.
- bool czy_double (const string &przelacznik_zmiennoprzecinkowy)

  Funkcja sprawdzająca, czy podany ciąg znakowy zawiera liczbę zmiennoprzecinkową.
- void konwertuj_przelaczniki (const string &pe, const string &ka, const string &wu, const string &er, unsigned int &p, unsigned int &k, double &w, double &r)

  Funkcja konwertująca przełączniki będące liczbami z ciągów znakowych na odpowiedni typ.
- void dopisz_enter (const string &sciezka_pliku)

  Funkcja dopisująca znak nowej linii w pliku wejściowym.
- void naprawa_strumienia (ifstream &file_in)

  Funkcja doprowadzająca strumień plikowy do stanu umożliwiającego dalsze wczytywanie.
- void sprawdz_koniec (ifstream &file_in, const string &sciezka_pliku)

  Funkcja sprawdzająca czy w pliku, z którego wczytywane są dane, znajduje się na końcu pusta linia konieczna do prawidłowego wczytywania danych przy pomocy funkcji .eof() z biblioteki fstream.
- bool wczytaj_z_pliku (ifstream &file_in, list< list< int >> &populacja, const string &sciezka_pliku)

  Funkcja wczytująca dane z pliku wejściowego do listy reprezentującej populację.
- void losuj_pare (list< list< int >> &populacja, int &pierwszy, int &drugi, int &licznik_losowan)

  Funkcja losująca dwa osobniki do krzyżowania zgodnie z założeniami tematu.
- void znajdz_krzyzowany (const list< list< int >> &populacja, int &chromosom_krzyzowany, list< int > &chromosom_szukany)

  Funkcja odnajdująca w populacji chromosom, który ma być krzyżowany.
- list< int > pobierz_koncowke (list< list< int >> &populacja, int &miejsce_przerwania, list< int > &aktualny_chromosom)

  Funkcja zwracająca listę będącą częścią chromosomu.
- void usun_skopiowane (const int &miejsce_przerwania, list< int > &aktualny_chromosom)

  Funkcja usuwająca niepotrzebną część chromosomu.
- void scal_chromosomy (const list< int > &fragment_dolaczany, list< int > &aktualny_chromosom)

  Funkcja łącząca odpowiednie elementy chromosomów krzyżowanych.
- void zamien_z_kopia (list< list< int >> &populacja, list< int > &chromosom, const int &miejsce)

  Funkcja wpisująca do populacji osobniki po krzyżowaniu w miejsce tych z przed krzyżowania.
- void krzyzuj_pokolenie (list< list< int >> &populacja, const int &k)

  Funkcja wykonująca wszystkie krzyżowania w danym pokoleniu.
- void znajdz_najwieksza_srednia (const list< list< int >> &populacja, double &najwieksza_srednia)

  Funkcja wyliczająca najwyższą wartość funkcji dopasowania wśród osobników w populacji.
- void tworz_nowe_pokolenie (list< list< int >> &nowe_pokolenie, list< int > &osobnik, const double &srednia, const double &wsp_w, const double &wsp_r)

  Funkcja tworząca nowe pokolenie na podstawie wartości funkcji dopasowania osobników w populacji.
- void sprawdz_dopasowanie (const list< list< int >> &populacja, list< list< int >> &nowe_pokolenie, const double &wsp_w, const double &wsp_r)

  Funkcja licząca wartości dopasowania dla całego pokolenia.
- void konfrontuj_pokolenie_ze_srodowiskiem (list< list< int >> &populacja, const double &wsp_w, const double &wsp_r)

  Funkcja sprawdzająca dopasowania i zapisująca do populacji nowe pokolenie.
- void modyfikuj_przez_pokolenia (list< list< int >> &populacja, const int &p, const int &k, const double &w, const double &r)

  Funkcja wykonująca zamysł programu przez wszystkie zadane pokolenia.
- void wypisz_do_pliku (ofstream &file_out, const list< list< int >> &populacja)

  Funkcja wypisująca dane z populacji (po odpowiednich modyfikacjach) do pliku wyjściowego.

3.3.1 Dokumentacja funkcji

3.3.1.1 czy_double()

```
bool czy_double (
            const string & przelacznik_zmiennoprzecinkowy )
```

Funkcja sprawdzająca, czy podany ciąg znakowy zawiera liczbę zmiennoprzecinkową.

Funkcja działa w oparciu o dwie pętle przebiegające przez ciąg znakowy.
Jedna zlicza ilość przecinków w ciągu, a druga sprawdza czy każdy znak jest liczbą lub przecinkiem.
Jeżeli ilość przecinków w liczbie była różna od 1 lub jakiś znak nie był ani cyfrą ani przecinkiem, to funkcja przyjmuje wartość fałsz.
Jest wywoływana do sprawdzenia czy użytkownik podał przełączniki w prawidłowy sposób.

Zobacz również

czy_int()

Parametry

przelacznik_zmiennoprzecinkowy	Łańcuh znakowy zawierający jedną ze zmiennych używanych w zadaniu, która MUSI być liczbą zmiennoprzecinkową.

Zwraca

prawda jeśli przełącznik został podany prawidłowo lub fałsz jeśli nie.

3.3.1.2 czy_int()

```
bool czy_int (
            const string & przelacznik_calkowitoliczbowy )
```

Funkcja sprawdzająca, czy podany ciąg znaków zawiera liczbę całkowitą.

Funkcja działa w oparciu o pętlę przebiegającą przez ciąg znakowy i sprawdzającą, czy każdy znak jest cyfrą.
Jeśli natrafi na znak nie spełniający tego warunku to funkcja przyjmuje wartość fałsz. Jest wywoływana do sprawdzenia czy użytkownik podał przełączniki w prawidłowy sposób.

Zobacz również

czy_double()

Parametry

przelacznik_calkowitoliczbowy	ciąg znakowy zawierający jedną ze zmiennych używanych w zadaniu, która MUSI być liczbą całkowitą.

Zwraca

prawda jeśli przełącznik został odpowiednio podany lub fałsz jeśli nie.

3.3.1.3 dopisz_enter()

```
void dopisz_enter (
            const string & sciezka_pliku )
```

Funkcja dopisująca znak nowej linii w pliku wejściowym.

Działanie można opisać następująco: otwiera plik o ścieżce podanej w argumentach jako strumień wyjściowy, dopisuje znak końca linii, a następnie zamyka plik.
Nie ingeruje w inne wartości podane w pliku przez użytkownika.
Jest wywoływana, gdy funkcja sprawdz_koniec() wykryje, że w pliku brak na końcu pustej linii, która jest konieczna do prawidłowego działania funkcji .eof() z biblioteki fstream, przy pomocy której wczytywane są dane z pliku.
Nie zwraca żadnej wartości.

Parametry

sciezka_pliku	zmienna będąca ciągiem znaków zawierająca ścieżkę do pliku, w którym ma być dokonana zmiana.

3.3.1.4 konfrontuj_pokolenie_ze_srodowiskiem()

```
void konfrontuj_pokolenie_ze_srodowiskiem (
            list< list< int >> & populacja,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja sprawdzająca dopasowania i zapisująca do populacji nowe pokolenie.

Tworzy lokalną zmienną, dla której wywołuje funkcję sprawdz_dopasowanie(). Po uzyskaniu wyników posiada już nowe pokolenie po dokonaniu zmian opisanych zadaniem, więc czyści starą populację i zapisuje do niej nowe pokolenie uzyskane z konfrontacji ze środowiskiem.
Jest wywoływana przez funkcję modyfikuj_przez_pokolenia na etapie, w którym zostały już wykonane konieczne krzyżowania.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca populację osobników rozwijaną zgodnie z założeniami zadania.

Parametry

wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji sprawdz_dopasowanie().
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji sprawdz_dopasowanie().

3.3.1.5 konwertuj_przelaczniki()

```
void konwertuj_przelaczniki (
            const string & pe,
            const string & ka,
            const string & wu,
            const string & er,
            unsigned int & p,
            unsigned int & k,
            double & w,
            double & r )
```

Funkcja konwertująca przełączniki będące liczbami z ciągów znakowych na odpowiedni typ.

Działa w oparciu o funkcje stoi oraz stod.
Jest wywoływana po sprawdzeniu czy przełączniki są zapisane w poprawny sposób.
Nie zwraca żadnej wartości.

Parametry

pe	zmienna będąca ciągiem znaków zawierająca przełącznik "-p".
ka	zmienna będąca ciągiem znaków zawierająca przełącznik "-k".
wu	zmienna będąca ciągiem znaków zawierająca przełącznik "-w".
er	zmienna będąca ciągiem znaków zawierająca przełącznik "-r".
p	zmienna będąca nieujemną liczbą całkowitą, w której ma zostać zapisany przełącznik "-p" podany przez użytkownika.
k	zmienna będąca nieujemną liczbą całkowitą, w której ma zostać zapisany przełącznik "-k" podany przez użytkownika.
w	zmienna będąca liczbą zmiennoprzecinkową, w której ma zostać zapisany przełącznik "-w" podany przez użytkownika.
r	zmienna będąca liczbą zmiennoprzecinkową, w której ma zostać zapisany przełącznik "-r" podany przez użytkownika.

3.3.1.6 krzyzuj_pokolenie()

```
void krzyzuj_pokolenie (
            list< list< int >> & populacja,
            const int & k )
```

Funkcja wykonująca wszystkie krzyżowania w danym pokoleniu.

Operuje na pętli sprawdzającej ilość wykonanych krzyżowań. Wewnątrz tej pętli tworzone są zmienne lokalne, a wykonywane następnie instrukcje można opisać następująco:
losowanie obu chromosomów do krzyżowania, zapisanie kopii jednego z nich do zmiennej, kopiowanie danych za losowym miejscem w chromosomie do krzyżowania, usunięcie elementów już skopiowanych z chromosomu, wykonanie ostatnich trzech kroków dla drugiego chromosomu, scalenie początku jednego chromosomu z końcem drugiego i odwrotnie, wstawienie kopii (skrzyżowane chromosomy) do edytowanej populacji w miejsce tych pobranych na początku.
Nie zwraca żadnej wartości.
Jest wywoływana przez funkcję modyfikuj_przez_pokolenia wykonującą założenia programu.

Nota

Wszystkie działania wymienione po przecinku odbywają się przy pomocy zagnieżdzonych funkcji, których dokumentację umieszczono w sekcji "zobacz również".

Zobacz również

losuj_pare(), znajdz_krzyzowany(), pobierz_koncowke(), usun_skopiowane(), scal_chromosomy(), zamien_z_kopia().

Parametry

populacja	lista reprezentująca populację, której osobniki mają być krzyżowane.
k	zmienna będąca liczbą całkowitą przechowująca ilość krzyżowań zadanych przez użytkownika do wykonania w każdym pokoleniu.

3.3.1.7 losuj_pare()

```
void losuj_pare (
            list< list< int >> & populacja,
            int & pierwszy,
            int & drugi,
            int & licznik_losowan )
```

Funkcja losująca dwa osobniki do krzyżowania zgodnie z założeniami tematu.

Funkcja losuje dwie liczby z przedziału będącego odbiciem ilości osobników w populacji, a następnie inkrementuje licznik krzyżowań w pokoleniu.
Zawiera ochronę przed wylosowaniem danego osobnika do krzyżowania z nim samym.
Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu.
Nie zwraca wartości.

Parametry

populacja	lista reprezentująca populację, z której mają być losowane osobniki.
pierwszy	zmienna będąca liczbą całkowitą, w której będzie przechowywany indeks jednego z chromosomów wybranych do krzyżowania.
drugi	zmienna będąca liczbą całkowitą, w której będzie przechowywany indeks drugiego z chromosomów wybranych do krzyżowania.
licznik_losowan	zmienna będąca liczbą całkowitą, która przechowuje ilość krzyżowań dokonanych w pokoleniu.

3.3.1.8 modyfikuj_przez_pokolenia()

```
void modyfikuj_przez_pokolenia (
            list< list< int >> & populacja,
            const int & p,
            const int & k,
            const double & w,
            const double & r )
```

Funkcja wykonująca zamysł programu przez wszystkie zadane pokolenia.

Funkcja używa pętli iterującej przez pokolenia wewnątrz której wywoływane są funkcje odpowiednialne za krzyżowanie oraz testowanie dopasowania, a więc krzyzuj_pokolenie() oraz konfrontuj_pokolenie_ze_srodowiskiem().
Jest wywoływana w funkcji main jako funkcja odpowiedzialna za wykonanie całego zadania.
Nie zwraca żadnej wartości.

Nota

Właściwie stworzona głównie po to, aby zminimalizować ilość kodu w funkcji main odpowiedzialnego za implementację warunków zadania do jednej funkcji.

Parametry

populacja	lista reprezentująca populację osobników pobraną z pliku wejściowego.
p	zmienna będąca liczbą całkowitą przechowująca ilość pokoleń podaną przez użytkownika.
k	zmienna będąca liczbą całkowitą przechowująca ilość krzyżowań do wykonania w danych pokoleniu podaną przez użytkownika.
w	zmienna będąca liczbą zmiennoprzecinkową przechowująca współczynnik wymierania w.
r	zmienna będąca liczbą zmiennoprzecinkową przechowująca współczynnik rozmnażania r.

3.3.1.9 naprawa_strumienia()

```
void naprawa_strumienia (
            ifstream & file_in )
```

Funkcja doprowadzająca strumień plikowy do stanu umożliwiającego dalsze wczytywanie.

Funkcja sprawdza, czy stan podanego w argumentach strumienia jest niepoprawny. W razie potrzeby czyści flagi błędów oraz bufor.
Jest wywoływana, gdy funkcja sprawdz_koniec() wykryje, że w pliku znajduje się na końcu pusta linia konieczna do prawidłowego działania funkcji .eof() z biblioteki fstream, przy pomocy której wczytywane są dane z pliku. Konieczność naprawy strumienia wynika ze sposobu działania funkcji sprawdz_koniec().

Parametry

file_in	strumień wejściowy, z którego czytane były dane, a w którym wystąpił błąd.

3.3.1.10 pobierz_koncowke()

```
list<int> pobierz_koncowke (
            list< list< int >> & populacja,
            int & miejsce_przerwania,
            list< int > & aktualny_chromosom )
```

Funkcja zwracająca listę będącą częścią chromosomu.

Funkcja losuje miejsce przerwania jednego z chromosomów krzyżowanych i pobiera do listy wartości genów tego chromosomu za miejscem przerwania, aby można je było później dołączyć do drugiego krzyżowanego chromosomu.
Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu po "uzyskaniu dostępu" do konkretnego chromosomu (funkcją znajdz_krzyzowany())

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, w której znajduje się chromosom przerywany.
miejsce_przerwania	zmienna będąca liczbą całkowitą, w której będzie przechowywana informacja na temat miejsca przerwania chromosomu.
aktualny_chromosom	lista reprezentująca chromosom osobnika, który będzie przerywany.

Zwraca

Fragment chromosomu, który ma być przeniesiony w skutek krzyżowania.

3.3.1.11 scal_chromosomy()

```
void scal_chromosomy (
            const list< int > & fragment_dolaczany,
            list< int > & aktualny_chromosom )
```

Funkcja łącząca odpowiednie elementy chromosomów krzyżowanych.

Funkcja dołącza do chromosomu, który przyjęła jako argument fragment drugiego.

Uwaga

Cała operacja odbywa się na elementach odpowiednio przygotowanych innymi funkcjami (znajdz_krzyzowany(), pobierz_koncowke(), usun_skopiowane()). Jest wywoływana przez funkcję krzyżującą osobniki na etapie posiadania w pamięci elementów chromosomów, które mają być połączone.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

fragment_dolaczany	fragment pobrany z końca chromosomu jednego osobnika w celu dołączenia do drugiego.
aktualny_chromosom	lista reprezentująca chromosom osobnika, do którego dodane mają być dane w wyniku krzyżowania.

3.3.1.12 sprawdz_dopasowanie()

```
void sprawdz_dopasowanie (
            const list< list< int >> & populacja,
            list< list< int >> & nowe_pokolenie,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja licząca wartości dopasowania dla całego pokolenia.

Po utworzeniu zmiennych lokalnych wywołuje funkcje znajdz_najwieksza_srednia() w celu umożliwienia sobie liczenia wartości funkcji dopasowania dla wszystkich elementów. Następnie przebiega przy pomocy pętli przez populację i liczy wartości funkcji dopasowania dla kolejnych osobników, po drodze przypisując wartością większym od jeden wartość 1 (dla usystematyzowania przedziału wartości) i dla każdego osobnika wywołuje funkcję tworz_nowe_pokolenie() tym samym budując stopniowo nowe pokolenie populacji.
Jest wywoływana przez funkcje konfrontuj_pokolenie_ze_srodowiskiem zaraz po utworzeniu zmiennych lokalnych.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca pokolenie osobników rozwijaną zgodnie z założeniami zadania.
nowe_pokolenie	lista reprezentująca nowe pokolenie tworzone na podstawie populacji wejściowej.
wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji tworz_nowe_pokolenie().
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji tworz_nowe_pokolenie().

3.3.1.13 sprawdz_koniec()

```
void sprawdz_koniec (
            ifstream & file_in,
            const string & sciezka_pliku )
```

Funkcja sprawdzająca czy w pliku, z którego wczytywane są dane, znajduje się na końcu pusta linia konieczna do prawidłowego wczytywania danych przy pomocy funkcji .eof() z biblioteki fstream.

Funkcja najpierw zmienia miejsce wczytywania na -2 w stosunku do końca pliku, a następnie wczytuje jeden znak. Jeśli natrafi na liczbę to wywołuje funkcję dopisz_enter(sciezka_pliku), a jeśli na znak końca pliku, to funkcje naprawa_strumienia(file_in).
Jest wywoływana przed rozpoczęciem wczytywania danych z pliku.

Parametry

file_in	strumień wejściowy, który ma być poddany sprawdzeniu.
sciezka_pliku	zmienna będąca ciągiem znaków zawierająca ścieżkę do pliku, który ma być poddany sprawdzeniu.

3.3.1.14 tworz_nowe_pokolenie()

```
void tworz_nowe_pokolenie (
            list< list< int >> & nowe_pokolenie,
            list< int > & osobnik,
            const double & srednia,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja tworząca nowe pokolenie na podstawie wartości funkcji dopasowania osobników w populacji.

Funkcja porównuje wartość funkcji dopasowania podanego jej poprzez argumenty osobnika z wartościami współczynników "w" oraz "r" podanymi przez użytkownika, a następnie w razie potrzeby dodaje osobnika (raz lub dwa razy) do listy reprezentującej nowe pokolenie populacji po wykonaniu krzyżowań dla danego pokolenia.
Jest wywoływana przez funkcję sprawdz_dopasowanie() po wyliczeniu wartości funkcji dopasowania dla konkretnego osobnika.
Nie zwraca żadnej wartości.

Parametry

nowe_pokolenie	lista reprezentująca pokolenie osobników tworzone zgodnie z założeniami zadania.
osobnik	lista reprezentująca chromosom osobnika poddawanego testom.
srednia	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość funkcji dopasowania osobnika poddawanego testom.
wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika.
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika.

Uwaga

Po wykonaniu funkcji zmienna przyjęta jako osobnik przechowuje populację zmodyfikowaną zgodnie ze wszystkimi założeniami zadania dla jednego pokolenia.

3.3.1.15 usun_skopiowane()

```
void usun_skopiowane (
            const int & miejsce_przerwania,
            list< int > & aktualny_chromosom )
```

Funkcja usuwająca niepotrzebną część chromosomu.

Funkcja usuwa wszystkie elementy za miejscem przerwania (znalezionym w funkcji pobierz_koncowke()) z chromosomu, który ma być krzyżowany, po to, aby móc w ich miejsce wpisać część drugiego z pary krzyżowanych chromosomów.
Jest wywoływana przez funkcję krzyżująca osobniki w pokoleniu po skopiowaniu danych za miejscem przerwania funkcją pobierz_koncowke().

Nota

Jest wywoływana dwa razy dla każdego krzyżowania! Nie zwraca wartości.

Parametry

miejsce_przerwania	zmienna będąca liczbą całkowitą, w której przechowywana jest informacja na temat miejsca przerwania chromosomu.
aktualny_chromosom	chromosom osobnika, który ma być poddany operacji.

3.3.1.16 wczytaj_z_pliku()

```
bool wczytaj_z_pliku (
            ifstream & file_in,
            list< list< int >> & populacja,
            const string & sciezka_pliku )
```

Funkcja wczytująca dane z pliku wejściowego do listy reprezentującej populację.

Działanie można omówić następująco:
Funkcja tworzy odpowiednie zmienne lokalne i sprawdza format pliku wejściowego funkcją sprawdz_koniec() (która go modyfikuje w razie potrzeby przy pomocy innych funkcji).
Następnie dopóki plik się nie skończył kolejno: pobiera jedną linię, zamienia ją na strumień, wczytuje ze strumienia (w razie napotkania w nim litery przyjmuje wartość fałsz i kończy działanie) i zapisuje do populacji po drodze usuwając zbędne elementy pojawiające się w wyniku stosowania w pętlach funkcji .eof() z bibliotek fstream oraz sstream.
Jeśli całość przebiegła pomyślnie przyjmuje wartość prawda.
Jest wywoływana po sprawdzeniu czy pliki podane przez użytkownika zostały prawidłowo otwarte.

Parametry

file_in	strumień wejściowy, z którego mają być wczytywane dane.
populacja	lista reprezentująca populację, do której mają być wczytywane dane.
sciezka_pliku	ścieżka do pliku, z którego wczytywane są dane (do podania do funkcji sprawdz_koniec())

Zwraca

prawda jeśli operacja wczytania przebiegła prawidłowo lub fałsz jeśli po drodze wystąpiły błędy.

3.3.1.17 wypisz_do_pliku()

```
void wypisz_do_pliku (
            ofstream & file_out,
            const list< list< int >> & populacja )
```

Funkcja wypisująca dane z populacji (po odpowiednich modyfikacjach) do pliku wyjściowego.

Używa pętli zagnieżdżonej w pętli do wypisywania chromosomów osobników populacji w formie zgodnej z założeniami zadania.
Jest wywoływana po wszystkich modyfikacjach populacji.
Nie zwraca żadnej wartości.

3.3.1.18 zamien_z_kopia()

```
void zamien_z_kopia (
            list< list< int >> & populacja,
            list< int > & chromosom,
            const int & miejsce )
```

Funkcja wpisująca do populacji osobniki po krzyżowaniu w miejsce tych z przed krzyżowania.

Dokładnie rzecz biorąc funkcja zamienia w populacji listy osobników wybranych do losowania z listami lokalnymi, które w chwili wywołania funkcji zawierają już dane osobników po dokonaniu krzyżowania.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, na której dokonywane są zmiany.
chromosom	lista reprezentująca osobnika, który będzie wpisany do populacji.
zmienna	będąca liczbą całkowitą, która przechowuje indeks jednego z chromosomów wybranych do krzyżowania.

3.3.1.19 znajdz_krzyzowany()

```
void znajdz_krzyzowany (
            const list< list< int >> & populacja,
            int & chromosom_krzyzowany,
            list< int > & chromosom_szukany )
```

Funkcja odnajdująca w populacji chromosom, który ma być krzyżowany.

Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu po wylosowaniu, które to mają być (funkcją losuj_pare()).
Nie zwraca wartości.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, w której szukamy chromosomu.
chromosom_krzyzowany	indeks chromosomu w populacji, który ma być znaleziony.
chromosom_szukany	lista reprezentująca osobnika, który ma być znaleziony.

3.3.1.20 znajdz_najwieksza_srednia()

```
void znajdz_najwieksza_srednia (
            const list< list< int >> & populacja,
            double & najwieksza_srednia )
```

Funkcja wyliczająca najwyższą wartość funkcji dopasowania wśród osobników w populacji.

Funkcja liczy wartości funkcji dopasowania dla osobników w populacji i zapisuje największa z nich do zmiennej. Jest wywoływana przez funkcję sprawdz_dopasowanie() w celu umożliwienia jej liczenia wartości funkcji dopasowania dla wszystkich elementów.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca populację osobników, której osobniki mają wyliczaną wartość funkcji dopasowania.
najwieksza_srednia	zmienna będąca liczbą zmiennoprzecinkową, która ma przechowywać najwyższą wartość funkcji dopasowania.

3.4 Dokumentacja pliku MODEL.h

```
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>
#include <list>
#include <ctime>
#include <cstdlib>
```

Funkcje

- bool czy_int (const string &przelacznik_calkowitoliczbowy)

  Funkcja sprawdzająca, czy podany ciąg znaków zawiera liczbę całkowitą.
- bool czy_double (const string &przelacznik_zmiennoprzecinkowy)

  Funkcja sprawdzająca, czy podany ciąg znakowy zawiera liczbę zmiennoprzecinkową.
- void konwertuj_przelaczniki (const string &pe, const string &ka, const string &wu, const string &er, unsigned int &p, unsigned int &k, double &w, double &r)

Funkcja konwertująca przełączniki będące liczbami z ciągów znakowych na odpowiedni typ.

- void dopisz_enter (const string &sciezka_pliku)

  Funkcja dopisująca znak nowej linii w pliku wejściowym.
- void naprawa_strumienia (ifstream &file_in)

  Funkcja doprowadzająca strumień plikowy do stanu umożliwiającego dalsze wczytywanie.
- void sprawdz_koniec (ifstream &file_in, const string &sciezka_pliku)

  Funkcja sprawdzająca czy w pliku, z którego wczytywane są dane, znajduje się na końcu pusta linia konieczna do prawidłowego wczytywania danych przy pomocy funkcji .eof() z biblioteki fstream.
- bool wczytaj_z_pliku (ifstream &file_in, list< list< int >> &populacja, const string &sciezka_pliku)

  Funkcja wczytująca dane z pliku wejściowego do listy reprezentującej populację.
- void losuj_pare (list< list< int >> &populacja, int &pierwszy, int &drugi, int &licznik_losowan)

  Funkcja losująca dwa osobniki do krzyżowania zgodnie z założeniami tematu.
- void znajdz_krzyzowany (const list< list< int >> &populacja, int &chromosom_krzyzowany, list< int > &chromosom_szukany)

  Funkcja odnajdująca w populacji chromosom, który ma być krzyżowany.
- list< int > pobierz_koncowke (list< list< int >> &populacja, int &miejsce_przerwania, list< int > &aktualny_chromosom)

  Funkcja zwracająca listę będącą częścią chromosomu.
- void usun_skopiowane (const int &miejsce_przerwania, list< int > &aktualny_chromosom)

  Funkcja usuwająca niepotrzebną część chromosomu.
- void scal_chromosomy (const list< int > &fragment_dolaczany, list< int > &aktualny_chromosom)

  Funkcja łącząca odpowiednie elementy chromosomów krzyżowanych.
- void zamien_z_kopia (list< list< int >> &populacja, list< int > &chromosom, const int &miejsce)

  Funkcja wpisująca do populacji osobniki po krzyżowaniu w miejsce tych z przed krzyżowania.
- void krzyzuj_pokolenie (list< list< int >> &populacja, const int &k)

  Funkcja wykonująca wszystkie krzyżowania w danym pokoleniu.
- void znajdz_najwieksza_srednia (const list< list< int >> &populacja, double &najwieksza_srednia)

  Funkcja wyliczająca najwyższą wartość funkcji dopasowania wśród osobników w populacji.
- void tworz_nowe_pokolenie (list< list< int >> &nowe_pokolenie, list< int > &osobnik, const double &srednia, const double &wsp_w, const double &wsp_r)

  Funkcja tworząca nowe pokolenie na podstawie wartości funkcji dopasowania osobników w populacji.
- void sprawdz_dopasowanie (const list< list< int >> &populacja, list< list< int >> &nowe_pokolenie, const double &wsp_w, const double &wsp_r)

  Funkcja licząca wartości dopasowania dla całego pokolenia.
- void konfrontuj_pokolenie_ze_srodowiskiem (list< list< int >> &populacja, const double &wsp_w, const double &wsp_r)

  Funkcja sprawdzająca dopasowania i zapisująca do populacji nowe pokolenie.
- void modyfikuj_przez_pokolenia (list< list< int >> &populacja, const int &p, const int &k, const double &w, const double &r)

  Funkcja wykonująca zamysł programu przez wszystkie zadane pokolenia.
- void wypisz_do_pliku (ofstream &file_out, const list< list< int >> &populacja)

  Funkcja wypisująca dane z populacji (po odpowiednich modyfikacjach) do pliku wyjściowego.

3.4.1 Dokumentacja funkcji

3.4.1.1 czy_double()

```
bool czy_double (
            const string & przelacznik_zmiennoprzecinkowy )
```

Funkcja sprawdzająca, czy podany ciąg znakowy zawiera liczbę zmiennoprzecinkową.

Funkcja działa w oparciu o dwie pętle przebiegające przez ciąg znakowy.
Jedna zlicza ilość przecinków w ciągu, a druga sprawdza czy każdy znak jest liczbą lub przecinkiem.
Jeżeli ilość przecinków w liczbie była różna od 1 lub jakiś znak nie był ani cyfrą ani przecinkiem, to funkcja przyjmuje wartość fałsz.
Jest wywoływana do sprawdzenia czy użytkownik podał przełączniki w prawidłowy sposób.

Zobacz również

czy_int()

Parametry

przelacznik_zmiennoprzecinkowy	Łańcuh znakowy zawierający jedną ze zmiennych używanych w zadaniu, która MUSI być liczbą zmiennoprzecinkową.

Zwraca

prawda jeśli przełącznik został podany prawidłowo lub fałsz jeśli nie.

3.4.1.2 czy_int()

```
bool czy_int (
            const string & przelacznik_calkowitoliczbowy )
```

Funkcja sprawdzająca, czy podany ciąg znaków zawiera liczbę całkowitą.

Funkcja działa w oparciu o pętlę przebiegającą przez ciąg znakowy i sprawdzającą, czy każdy znak jest cyfrą.
Jeśli natrafi na znak nie spełniający tego warunku to funkcja przyjmuje wartość fałsz. Jest wywoływana do sprawdzenia czy użytkownik podał przełączniki w prawidłowy sposób.

Zobacz również

czy_double()

Parametry

przelacznik_calkowitoliczbowy	ciąg znakowy zawierający jedną ze zmiennych używanych w zadaniu, która MUSI być liczbą całkowitą.

Zwraca

prawda jeśli przełącznik został odpowiednio podany lub fałsz jeśli nie.

3.4.1.3 dopisz_enter()

```
void dopisz_enter (
            const string & sciezka_pliku )
```

Funkcja dopisująca znak nowej linii w pliku wejściowym.

Działanie można opisać następująco: otwiera plik o ścieżce podanej w argumentach jako strumień wyjściowy, dopisuje znak końca linii, a następnie zamyka plik.
Nie ingeruje w inne wartości podane w pliku przez użytkownika.
Jest wywoływana, gdy funkcja sprawdz_koniec() wykryje, że w pliku brak na końcu pustej linii, która jest konieczna do prawidłowego działania funkcji .eof() z biblioteki fstream, przy pomocy której wczytywane są dane z pliku.
Nie zwraca żadnej wartości.

Parametry

sciezka_pliku	zmienna będąca ciągiem znaków zawierająca ścieżkę do pliku, w którym ma być dokonana zmiana.

3.4.1.4 konfrontuj_pokolenie_ze_srodowiskiem()

```
void konfrontuj_pokolenie_ze_srodowiskiem (
            list< list< int >> & populacja,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja sprawdzająca dopasowania i zapisująca do populacji nowe pokolenie.

Tworzy lokalną zmienną, dla której wywołuje funkcję sprawdz_dopasowanie(). Po uzyskaniu wyników posiada już nowe pokolenie po dokonaniu zmian opisanych zadaniem, więc czyści starą populację i zapisuje do niej nowe pokolenie uzyskane z konfrontacji ze środowiskiem.
Jest wywoływana przez funkcję modyfikuj_przez_pokolenia na etapie, w którym zostały już wykonane konieczne krzyżowania.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca populację osobników rozwijaną zgodnie z założeniami zadania.
wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji sprawdz_dopasowanie().
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji sprawdz_dopasowanie().

3.4.1.5 konwertuj_przelaczniki()

```
void konwertuj_przelaczniki (
            const string & pe,
            const string & ka,
            const string & wu,
            const string & er,
            unsigned int & p,
            unsigned int & k,
            double & w,
            double & r )
```

Funkcja konwertująca przełączniki będące liczbami z ciągów znakowych na odpowiedni typ.

Działa w oparciu o funkcje stoi oraz stod.
Jest wywoływana po sprawdzeniu czy przełączniki są zapisane w poprawny sposób.
Nie zwraca żadnej wartości.

Parametry

pe	zmienna będąca ciągiem znaków zawierająca przełącznik "-p".
ka	zmienna będąca ciągiem znaków zawierająca przełącznik "-k".
wu	zmienna będąca ciągiem znaków zawierająca przełącznik "-w".
er	zmienna będąca ciągiem znaków zawierająca przełącznik "-r".
p	zmienna będąca nieujemną liczbą całkowitą, w której ma zostać zapisany przełącznik "-p" podany przez użytkownika.
k	zmienna będąca nieujemną liczbą całkowitą, w której ma zostać zapisany przełącznik "-k" podany przez użytkownika.
w	zmienna będąca liczbą zmiennoprzecinkową, w której ma zostać zapisany przełącznik "-w" podany przez użytkownika.
r	zmienna będąca liczbą zmiennoprzecinkową, w której ma zostać zapisany przełącznik "-r" podany przez użytkownika.

3.4.1.6 krzyzuj_pokolenie()

```
void krzyzuj_pokolenie (
            list< list< int >> & populacja,
            const int & k )
```

Funkcja wykonująca wszystkie krzyżowania w danym pokoleniu.

Operuje na pętli sprawdzającej ilość wykonanych krzyżowań. Wewnątrz tej pętli tworzone są zmienne lokalne, a wykonywane następnie instrukcje można opisać następująco:
losowanie obu chromosomów do krzyżowania, zapisanie kopii jednego z nich do zmiennej, kopiowanie danych za losowym miejscem w chromosomie do krzyżowania, usunięcie elementów już skopiowanych z chromosomu, wykonanie ostatnich trzech kroków dla drugiego chromosomu, scalenie początku jednego chromosomu z końcem drugiego i odwrotnie, wstawienie kopii (skrzyżowane chromosomy) do edytowanej populacji w miejsce tych pobranych na początku.
Nie zwraca żadnej wartości.
Jest wywoływana przez funkcję modyfikuj_przez_pokolenia wykonującą założenia programu.

Nota

Wszystkie działania wymienione po przecinku odbywają się przy pomocy zagnieżdzonych funkcji, których dokumentację umieszczono w sekcji "zobacz również".

Zobacz również

losuj_pare(), znajdz_krzyzowany(), pobierz_koncowke(), usun_skopiowane(), scal_chromosomy(), zamien_z_kopia().

Parametry

populacja	lista reprezentująca populację, której osobniki mają być krzyżowane.
k	zmienna będąca liczbą całkowitą przechowująca ilość krzyżowań zadanych przez użytkownika do wykonania w każdym pokoleniu.

3.4.1.7 losuj_pare()

```
void losuj_pare (
            list< list< int >> & populacja,
            int & pierwszy,
            int & drugi,
            int & licznik_losowan )
```

Funkcja losująca dwa osobniki do krzyżowania zgodnie z założeniami tematu.

Funkcja losuje dwie liczby z przedziału będącego odbiciem ilości osobników w populacji, a następnie inkrementuje licznik krzyżowań w pokoleniu.
Zawiera ochronę przed wylosowaniem danego osobnika do krzyżowania z nim samym.
Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu.
Nie zwraca wartości.

Parametry

populacja	lista reprezentująca populację, z której mają być losowane osobniki.
pierwszy	zmienna będąca liczbą całkowitą, w której będzie przechowywany indeks jednego z chromosomów wybranych do krzyżowania.
drugi	zmienna będąca liczbą całkowitą, w której będzie przechowywany indeks drugiego z chromosomów wybranych do krzyżowania.
licznik_losowan	zmienna będąca liczbą całkowitą, która przechowuje ilość krzyżowań dokonanych w pokoleniu.

3.4.1.8 modyfikuj_przez_pokolenia()

```
void modyfikuj_przez_pokolenia (
            list< list< int >> & populacja,
```

```
            const int & p,
            const int & k,
            const double & w,
            const double & r )
```

Funkcja wykonująca zamysł programu przez wszystkie zadane pokolenia.

Funkcja używa pętli iterującej przez pokolenia wewnątrz której wywoływane są funkcje odpowiednialne za krzyżowanie oraz testowanie dopasowania, a więc krzyzuj_pokolenie() oraz konfrontuj_pokolenie_ze_srodowiskiem().
Jest wywoływana w funkcji main jako funkcja odpowiedzialna za wykonanie całego zadania.
Nie zwraca żadnej wartości.

Nota

Właściwie stworzona głównie po to, aby zminimalizować ilość kodu w funkcji main odpowiedzialnego za implementację warunków zadania do jednej funkcji.

Parametry

populacja	lista reprezentująca populację osobników pobraną z pliku wejściowego.
p	zmienna będąca liczbą całkowitą przechowująca ilość pokoleń podaną przez użytkownika.
k	zmienna będąca liczbą całkowitą przechowująca ilość krzyżowań do wykonania w danych pokoleniu podaną przez użytkownika.
w	zmienna będąca liczbą zmiennoprzecinkową przechowująca współczynnik wymierania w.
r	zmienna będąca liczbą zmiennoprzecinkową przechowująca współczynnik rozmnażania r.

3.4.1.9 naprawa_strumienia()

```
void naprawa_strumienia (
            ifstream & file_in )
```

Funkcja doprowadzająca strumień plikowy do stanu umożliwiającego dalsze wczytywanie.

Funkcja sprawdza, czy stan podanego w argumentach strumienia jest niepoprawny. W razie potrzeby czyści flagi błędów oraz bufor.
Jest wywoływana, gdy funkcja sprawdz_koniec() wykryje, że w pliku znajduje się na końcu pusta linia konieczna do prawidłowego działania funkcji .eof() z biblioteki fstream, przy pomocy której wczytywane są dane z pliku. Konieczność naprawy strumienia wynika ze sposobu działania funkcji sprawdz_koniec().

Parametry

file_in	strumień wejściowy, z którego czytane były dane, a w którym wystąpił błąd.

3.4.1.10 pobierz_koncowke()

```
list<int> pobierz_koncowke (
```

```
            list< list< int >> & populacja,
            int & miejsce_przerwania,
            list< int > & aktualny_chromosom )
```

Funkcja zwracająca listę będącą częścią chromosomu.

Funkcja losuje miejsce przerwania jednego z chromosomów krzyżowanych i pobiera do listy wartości genów tego chromosomu za miejscem przerwania, aby można je było później dołączyć do drugiego krzyżowanego chromosomu.
Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu po "uzyskaniu dostępu" do konkretnego chromosomu (funkcją znajdz_krzyzowany())

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, w której znajduje się chromosom przerywany.
miejsce_przerwania	zmienna będąca liczbą całkowitą, w której będzie przechowywana informacja na temat miejsca przerwania chromosomu.
aktualny_chromosom	lista reprezentująca chromosom osobnika, który będzie przerywany.

Zwraca

Fragment chromosomu, który ma być przeniesiony w skutek krzyżowania.

3.4.1.11 scal_chromosomy()

```
void scal_chromosomy (
            const list< int > & fragment_dolaczany,
            list< int > & aktualny_chromosom )
```

Funkcja łącząca odpowiednie elementy chromosomów krzyżowanych.

Funkcja dołącza do chromosomu, który przyjęła jako argument fragment drugiego.

Uwaga

Cała operacja odbywa się na elementach odpowiednio przygotowanych innymi funkcjami (znajdz_krzyzowany(), pobierz_koncowke(), usun_skopiowane()). Jest wywoływana przez funkcję krzyżującą osobniki na etapie posiadania w pamięci elementów chromosomów, które mają być połączone.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

fragment_dolaczany	fragment pobrany z końca chromosomu jednego osobnika w celu dołączenia do drugiego.
aktualny_chromosom	lista reprezentująca chromosom osobnika, do którego dodane mają być dane w wyniku krzyżowania.

3.4.1.12 sprawdz_dopasowanie()

```
void sprawdz_dopasowanie (
            const list< list< int >> & populacja,
            list< list< int >> & nowe_pokolenie,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja licząca wartości dopasowania dla całego pokolenia.

Po utworzeniu zmiennych lokalnych wywołuje funkcje znajdz_najwieksza_srednia() w celu umożliwienia sobie liczenia wartości funkcji dopasowania dla wszystkich elementów. Następnie przebiega przy pomocy pętli przez populację i liczy wartości funkcji dopasowania dla kolejnych osobników, po drodze przypisując wartością większym od jeden wartość 1 (dla usystematyzowania przedziału wartości) i dla każdego osobnika wywołuje funkcję tworz_nowe_pokolenie() tym samym budując stopniowo nowe pokolenie populacji.
Jest wywoływana przez funkcje konfrontuj_pokolenie_ze_srodowiskiem zaraz po utworzeniu zmiennych lokalnych.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca pokolenie osobników rozwijaną zgodnie z założeniami zadania.
nowe_pokolenie	lista reprezentująca nowe pokolenie tworzone na podstawie populacji wejściowej.
wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji tworz_nowe_pokolenie().
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika podawana, by umożliwić wywołanie funkcji tworz_nowe_pokolenie().

3.4.1.13 sprawdz_koniec()

```
void sprawdz_koniec (
            ifstream & file_in,
            const string & sciezka_pliku )
```

Funkcja sprawdzająca czy w pliku, z którego wczytywane są dane, znajduje się na końcu pusta linia konieczna do prawidłowego wczytywania danych przy pomocy funkcji .eof() z biblioteki fstream.

Funkcja najpierw zmienia miejsce wczytywania na -2 w stosunku do końca pliku, a następnie wczytuje jeden znak. Jeśli natrafi na liczbę to wywołuje funkcję dopisz_enter(sciezka_pliku), a jeśli na znak końca pliku, to funkcje naprawa_strumienia(file_in).
Jest wywoływana przed rozpoczęciem wczytywania danych z pliku.

Parametry

file_in	strumień wejściowy, który ma być poddany sprawdzeniu.
sciezka_pliku	zmienna będąca ciągiem znaków zawierająca ścieżkę do pliku, który ma być poddany sprawdzeniu.

3.4.1.14 tworz_nowe_pokolenie()

```
void tworz_nowe_pokolenie (
            list< list< int >> & nowe_pokolenie,
            list< int > & osobnik,
            const double & srednia,
            const double & wsp_w,
            const double & wsp_r )
```

Funkcja tworząca nowe pokolenie na podstawie wartości funkcji dopasowania osobników w populacji.

Funkcja porównuje wartość funkcji dopasowania podanego jej poprzez argumenty osobnika z wartościami współczynników "w" oraz "r" podanymi przez użytkownika, a następnie w razie potrzeby dodaje osobnika (raz lub dwa razy) do listy reprezentującej nowe pokolenie populacji po wykonaniu krzyżowań dla danego pokolenia.
Jest wywoływana przez funkcję sprawdz_dopasowanie() po wyliczeniu wartości funkcji dopasowania dla konkretnego osobnika.
Nie zwraca żadnej wartości.

Parametry

nowe_pokolenie	lista reprezentująca pokolenie osobników tworzone zgodnie z założeniami zadania.
osobnik	lista reprezentująca chromosom osobnika poddawanego testom.
srednia	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość funkcji dopasowania osobnika poddawanego testom.
wsp_w	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "w" podaną przez użytkownika.
wsp_r	zmienna będąca liczbą zmiennoprzecinkową przechowująca wartość współczynnika "r" podaną przez użytkownika.

Uwaga

Po wykonaniu funkcji zmienna przyjęta jako osobnik przechowuje populację zmodyfikowaną zgodnie ze wszystkimi założeniami zadania dla jednego pokolenia.

3.4.1.15 usun_skopiowane()

```
void usun_skopiowane (
            const int & miejsce_przerwania,
            list< int > & aktualny_chromosom )
```

Funkcja usuwająca niepotrzebną część chromosomu.

Funkcja usuwa wszystkie elementy za miejscem przerwania (znalezionym w funkcji pobierz_koncowke()) z chromosomu, który ma być krzyżowany, po to, aby móc w ich miejsce wpisać część drugiego z pary krzyżowanych chromosomów.
Jest wywoływana przez funkcję krzyżująca osobniki w pokoleniu po skopiowaniu danych za miejscem przerwania funkcją pobierz_koncowke().

Nota

Jest wywoływana dwa razy dla każdego krzyżowania! Nie zwraca wartości.

Parametry

miejsce_przerwania	zmienna będąca liczbą całkowitą, w której przechowywana jest informacja na temat miejsca przerwania chromosomu.
aktualny_chromosom	chromosom osobnika, który ma być poddany operacji.

3.4.1.16 wczytaj_z_pliku()

```
bool wczytaj_z_pliku (
            ifstream & file_in,
            list< list< int >> & populacja,
            const string & sciezka_pliku )
```

Funkcja wczytująca dane z pliku wejściowego do listy reprezentującej populację.

Działanie można omówić następująco:
Funkcja tworzy odpowiednie zmienne lokalne i sprawdza format pliku wejściowego funkcją sprawdz_koniec() (która go modyfikuje w razie potrzeby przy pomocy innych funkcji).
Następnie dopóki plik się nie skończył kolejno: pobiera jedną linię, zamienia ją na strumień, wczytuje ze strumienia (w razie napotkania w nim litery przyjmuje wartość fałsz i kończy działanie) i zapisuje do populacji po drodze usuwając zbędne elementy pojawiające się w wyniku stosowania w pętlach funkcji .eof() z bibliotek fstream oraz sstream.
Jeśli całość przebiegła pomyślnie przyjmuje wartość prawda.
Jest wywoływana po sprawdzeniu czy pliki podane przez użytkownika zostały prawidłowo otwarte.

Parametry

file_in	strumień wejściowy, z którego mają być wczytywane dane.
populacja	lista reprezentująca populację, do której mają być wczytywane dane.
sciezka_pliku	ścieżka do pliku, z którego wczytywane są dane (do podania do funkcji sprawdz_koniec())

Zwraca

prawda jeśli operacja wczytania przebiegła prawidłowo lub fałsz jeśli po drodze wystąpiły błędy.

3.4.1.17 wypisz_do_pliku()

```
void wypisz_do_pliku (
            ofstream & file_out,
            const list< list< int >> & populacja )
```

Funkcja wypisująca dane z populacji (po odpowiednich modyfikacjach) do pliku wyjściowego.

Używa pętli zagnieżdżonej w pętli do wypisywania chromosomów osobników populacji w formie zgodnej z założeniami zadania.
Jest wywoływana po wszystkich modyfikacjach populacji.
Nie zwraca żadnej wartości.

3.4.1.18 zamien_z_kopia()

```
void zamien_z_kopia (
            list< list< int >> & populacja,
            list< int > & chromosom,
            const int & miejsce )
```

Funkcja wpisująca do populacji osobniki po krzyżowaniu w miejsce tych z przed krzyżowania.

Dokładnie rzecz biorąc funkcja zamienia w populacji listy osobników wybranych do losowania z listami lokalnymi, które w chwili wywołania funkcji zawierają już dane osobników po dokonaniu krzyżowania.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, na której dokonywane są zmiany.
chromosom	lista reprezentująca osobnika, który będzie wpisany do populacji.
zmienna	będąca liczbą całkowitą, która przechowuje indeks jednego z chromosomów wybranych do krzyżowania.

3.4.1.19 znajdz_krzyzowany()

```
void znajdz_krzyzowany (
            const list< list< int >> & populacja,
            int & chromosom_krzyzowany,
            list< int > & chromosom_szukany )
```

Funkcja odnajdująca w populacji chromosom, który ma być krzyżowany.

Jest wywoływana przez funkcję krzyżującą osobniki w pokoleniu po wylosowaniu, które to mają być (funkcją losuj_pare()).
Nie zwraca wartości.

Nota

Jest wywoływana dwa razy dla każdego krzyżowania!

Parametry

populacja	lista reprezentująca populację, w której szukamy chromosomu.
chromosom_krzyzowany	indeks chromosomu w populacji, który ma być znaleziony.
chromosom_szukany	lista reprezentująca osobnika, który ma być znaleziony.

3.4.1.20 znajdz_najwieksza_srednia()

```
void znajdz_najwieksza_srednia (
            const list< list< int >> & populacja,
            double & najwieksza_srednia )
```

Funkcja wyliczająca najwyższą wartość funkcji dopasowania wśród osobników w populacji.

Funkcja liczy wartości funkcji dopasowania dla osobników w populacji i zapisuje największa z nich do zmiennej. Jest wywoływana przez funkcję sprawdz_dopasowanie() w celu umożliwienia jej liczenia wartości funkcji dopasowania dla wszystkich elementów.
Nie zwraca żadnej wartości.

Parametry

populacja	lista reprezentująca populację osobników, której osobniki mają wyliczaną wartość funkcji dopasowania.
najwieksza_srednia	zmienna będąca liczbą zmiennoprzecinkową, która ma przechowywać najwyższą wartość funkcji dopasowania.

3.5 Dokumentacja pliku plik_wejsciowy.txt

3.6 Dokumentacja pliku plik_wyjsciowy.txt

3.7 Dokumentacja pliku README.md

3.8 Dokumentacja pliku Widok.cpp

```
#include "WIDOK.h"
#include "MODEL.h"
```

Funkcje

- void instrukcja_obslugi ()

  Funkcja wyswietlająca na ekranie instrukcję obsługi.
- bool ilosc_przelacznikow (const int &args)

  Funkcja sprawdzająca ilość przełączników podanych przez użytkownika podczas uruchamiania.
- bool pobierz_przelacznik (char ∗params[ ], const string &przelacznik, string &wyjscie)

  Funkcja znajdująca wśród podanych przełączników jeden konkretny i zapisująca go do zmiennej będącej ciągiem znaków.
- bool sprawdz_poprawnosc_przelacznikow (const string &pe, const string &ka, const string &wu, const string &er)

  Funkcja sprawdzająca czy przełączniki {"-w","-p","-k","-r"} wpisano w odpowiedniej postaci.
- bool sprawdz_otwarcie (ifstream &file_in, ofstream &file_out)

  Funkcja sprawdzająca czy pliki wejściowe zostały otwarte.
- void wypisz_populacje (list< list< int >> &lista_list)

  Funkcja wypisująca populację przechowywaną w liście na ekran.
- void litera_w_pliku ()

  Funkcja wyświetlająca na ekranie informację o błędzie w podanym pliku wejściowym.

3.8.1 Dokumentacja funkcji

3.8.1.1 ilosc_przelacznikow()

```
bool ilosc_przelacznikow (
            const int & args )
```

Funkcja sprawdzająca ilość przełączników podanych przez użytkownika podczas uruchamiania.

Jest wywoływana na początku programu w celu wykrycia ewentualnych błędów.

Parametry

args	ilość przełączników podanych w wierszu poleceń.

Zwraca

prawda jeśli podano odpowiednią ilość przełączników lub fałsz jeśli nie.

3.8.1.2 instrukcja_obslugi()

```
void instrukcja_obslugi ( )
```

Funkcja wyswietlająca na ekranie instrukcję obsługi.

Jest implementowana w razie otwarcia programu ze złą liczbą argumentów lub gdy przełączniki nie zostaną prawidłowo wczytane.
Nie przyjmuje parametrow.
Jest wywoływana we wszystkich momentach, w których program natrafia na błąd.
Nie zwraca wartości, lecz wypisuje na ekran.

3.8.1.3 litera_w_pliku()

```
void litera_w_pliku ( )
```

Funkcja wyświetlająca na ekranie informację o błędzie w podanym pliku wejściowym.

Stworzona dla odróżnienia od podstawowej instrukcji obsługi, gdyż dotyczy względnie szczególnego przypadku niezwiązanego bezpośrednio z podaniem argumentu wiersza poleceń, a z samym plikiem, który został poprawnie wczytany, lecz zawiera błąd.
Nie przyjmuje parametrów.
Nie zwraca wartości, lecz wypisuje na ekran.
Jest wywoływana, gdy program wykryje błąd strumienia wczytującego z pliku (inny niż ten związany z końcem).

Zobacz również

wczytaj_z_pliku();

3.8.1.4 pobierz_przelacznik()

```
bool pobierz_przelacznik (
            char * params[],
            const string & przelacznik,
            string & wyjscie )
```

Funkcja znajdująca wśród podanych przełączników jeden konkretny i zapisująca go do zmiennej będącej ciągiem znaków.

Jest wywoływana na początku programu w celu pobrania danych i wykrycia ewentualnych błędów.

Parametry

params[]	tablica przełączników pobrana z wiersza poleceń.
przelacznik	odpowiedni przelacznik ze zbioru {"-i","-o","-w","-p","-k","-r"}.
wyjscie	zmienna, w której zostanie zapisana ścieżka do odpowiedniego pliku.

Zwraca

prawda jeśli odpowiedni przełącznik został znaleziony lub fałsz jeśli nie.

3.8.1.5 sprawdz_otwarcie()

```
bool sprawdz_otwarcie (
            ifstream & file_in,
            ofstream & file_out )
```

Funkcja sprawdzająca czy pliki wejściowe zostały otwarte.

Jest wywoływana po utworzeniu strumieni plikowych ze ścieżkami, które podano tu jako argumenty.
Jeśli którykolwiek z plików nie został otwarty prawidłowo, wypisuje stosowną informację na ekran i przyjmuje wartość fałsz.
Jeśli wszystko jest w porządku przyjmuje wartość prawda.

Parametry

file_in	strumień wejściowy do sprawdzenia.
file_out	strumień wyjściowy do sprawdzenia

Zwraca

Prawda jeśli pliki zostały otwarte prawidłowo lub fałsz jeśli nie.

3.8.1.6 sprawdz_poprawnosc_przelacznikow()

```
bool sprawdz_poprawnosc_przelacznikow (
            const string & pe,
            const string & ka,
            const string & wu,
            const string & er )
```

Funkcja sprawdzająca czy przełączniki {"-w","-p","-k","-r"} wpisano w odpowiedniej postaci.

Funkcja sprawdza czy przełączniki "-k" oraz "-p" zostały podane przez użytkownika jako liczby całkowite dodatnie przy pomocy funkcji czy_int(), a także czy przełączniki "-w" oraz "-r" zostały podane przez użytkownika jako liczby zmiennoprzecinkowe przy pomocy funkcji czy_double().
Jest implementowana po pobraniu przełączników z wiersza poleceń, ale przed rozpoczęciem działań na nich.

Parametry

pe	zmienna będąca ciągiem znaków zawierająca przełącznik "-p"
ka	zmienna będąca ciągiem znaków zawierająca przełącznik "-k"
wu	zmienna będąca ciągiem znaków zawierająca przełącznik "-w"
er	zmienna będąca ciągiem znaków zawierająca przełącznik "-r"

Zwraca

prawda jeśli wszystkie przełączniki są zapisane prawidłowo lub fałsz jeśli któryś nie jest.

3.8.1.7 wypisz_populacje()

```
void wypisz_populacje (
            list< list< int >> & lista_list )
```

Funkcja wypisująca populację przechowywaną w liście na ekran.

Była używana przez piszącego program w celach testowych i pozwalała orientować się w tym, na jakim etapie jest program.

Ostrzeżenie

Obecnie nie jest nigdzie wywoływana i pozostawiono ją wyłącznie w razie chęci przetestowania programu przez oceniającego.

3.9 Dokumentacja pliku WIDOK.h

```
#include <iostream>
#include <string>
#include <list>
```

Funkcje

- void instrukcja_obslugi ()

  Funkcja wyswietlająca na ekranie instrukcję obsługi.
- bool ilosc_przelacznikow (const int &args)

  Funkcja sprawdzająca ilość przełączników podanych przez użytkownika podczas uruchamiania.
- bool pobierz_przelacznik (char ∗params[ ], const string &przelacznik, string &wyjscie)

  Funkcja znajdująca wśród podanych przełączników jeden konkretny i zapisująca go do zmiennej będącej ciągiem znaków.
- bool sprawdz_poprawnosc_przelacznikow (const string &pe, const string &ka, const string &wu, const string &er)

  Funkcja sprawdzająca czy przełączniki {"-w","-p","-k","-r"} wpisano w odpowiedniej postaci.
- bool sprawdz_otwarcie (ifstream &file_in, ofstream &file_out)

  Funkcja sprawdzająca czy pliki wejściowe zostały otwarte.
- void wypisz_populacje (list< list< int >> &lista_list)

  Funkcja wypisująca populację przechowywaną w liście na ekran.
- void litera_w_pliku ()

  Funkcja wyświetlająca na ekranie informację o błędzie w podanym pliku wejściowym.

3.9.1 Dokumentacja funkcji

3.9.1.1 ilosc_przelacznikow()

```
bool ilosc_przelacznikow (
            const int & args )
```

Funkcja sprawdzająca ilość przełączników podanych przez użytkownika podczas uruchamiania.

Jest wywoływana na początku programu w celu wykrycia ewentualnych błędów.

Parametry

args	ilość przełączników podanych w wierszu poleceń.

Zwraca

prawda jeśli podano odpowiednią ilość przełączników lub fałsz jeśli nie.

3.9.1.2 instrukcja_obslugi()

```
void instrukcja_obslugi ( )
```

Funkcja wyswietlająca na ekranie instrukcję obsługi.

Jest implementowana w razie otwarcia programu ze złą liczbą argumentów lub gdy przełączniki nie zostaną prawidłowo wczytane.
Nie przyjmuje parametrow.
Jest wywoływana we wszystkich momentach, w których program natrafia na błąd.
Nie zwraca wartości, lecz wypisuje na ekran.

3.9.1.3 litera_w_pliku()

```
void litera_w_pliku ( )
```

Funkcja wyświetlająca na ekranie informację o błędzie w podanym pliku wejściowym.

Stworzona dla odróżnienia od podstawowej instrukcji obsługi, gdyż dotyczy względnie szczególnego przypadku niezwiązanego bezpośrednio z podaniem argumentu wiersza poleceń, a z samym plikiem, który został poprawnie wczytany, lecz zawiera błąd.
Nie przyjmuje parametrów.
Nie zwraca wartości, lecz wypisuje na ekran.
Jest wywoływana, gdy program wykryje błąd strumienia wczytującego z pliku (inny niż ten związany z końcem).

Zobacz również

wczytaj_z_pliku();

3.9.1.4 pobierz_przelacznik()

```
bool pobierz_przelacznik (
            char * params[],
            const string & przelacznik,
            string & wyjscie )
```

Funkcja znajdująca wśród podanych przełączników jeden konkretny i zapisująca go do zmiennej będącej ciągiem znaków.

Jest wywoływana na początku programu w celu pobrania danych i wykrycia ewentualnych błędów.

Parametry

params[]	tablica przełączników pobrana z wiersza poleceń.
przelacznik	odpowiedni przelacznik ze zbioru {"-i","-o","-w","-p","-k","-r"}.
wyjscie	zmienna, w której zostanie zapisana ścieżka do odpowiedniego pliku.

Zwraca

prawda jeśli odpowiedni przełącznik został znaleziony lub fałsz jeśli nie.

3.9.1.5 sprawdz_otwarcie()

```
bool sprawdz_otwarcie (
            ifstream & file_in,
            ofstream & file_out )
```

Funkcja sprawdzająca czy pliki wejściowe zostały otwarte.

Jest wywoływana po utworzeniu strumieni plikowych ze ścieżkami, które podano tu jako argumenty.
Jeśli którykolwiek z plików nie został otwarty prawidłowo, wypisuje stosowną informację na ekran i przyjmuje wartość fałsz.
Jeśli wszystko jest w porządku przyjmuje wartość prawda.

Parametry

file_in	strumień wejściowy do sprawdzenia.
file_out	strumień wyjściowy do sprawdzenia

Zwraca

Prawda jeśli pliki zostały otwarte prawidłowo lub fałsz jeśli nie.

3.9.1.6 sprawdz_poprawnosc_przelacznikow()

```
bool sprawdz_poprawnosc_przelacznikow (
            const string & pe,
            const string & ka,
            const string & wu,
            const string & er )
```

Funkcja sprawdzająca czy przełączniki {"-w","-p","-k","-r"} wpisano w odpowiedniej postaci.

Funkcja sprawdza czy przełączniki "-k" oraz "-p" zostały podane przez użytkownika jako liczby całkowite dodatnie przy pomocy funkcji czy_int(), a także czy przełączniki "-w" oraz "-r" zostały podane przez użytkownika jako liczby zmiennoprzecinkowe przy pomocy funkcji czy_double().
Jest implementowana po pobraniu przełączników z wiersza poleceń, ale przed rozpoczęciem działań na nich.

Parametry

pe	zmienna będąca ciągiem znaków zawierająca przełącznik "-p"
ka	zmienna będąca ciągiem znaków zawierająca przełącznik "-k"
wu	zmienna będąca ciągiem znaków zawierająca przełącznik "-w"
er	zmienna będąca ciągiem znaków zawierająca przełącznik "-r"

Zwraca

prawda jeśli wszystkie przełączniki są zapisane prawidłowo lub fałsz jeśli któryś nie jest.

3.9.1.7 wypisz_populacje()

```
void wypisz_populacje (
            list< list< int >> & lista_list )
```

Funkcja wypisująca populację przechowywaną w liście na ekran.

Była używana przez piszącego program w celach testowych i pozwalała orientować się w tym, na jakim etapie jest program.

Ostrzeżenie

Obecnie nie jest nigdzie wywoływana i pozostawiono ją wyłącznie w razie chęci przetestowania programu przez oceniającego.

Indeks

czy_double
    Model.cpp, 7
    MODEL.h, 18
czy_int
    Model.cpp, 7
    MODEL.h, 19

Darwin.vcxproj.FileListAbsolute.txt, 5
dopisz_enter
    Model.cpp, 8
    MODEL.h, 20

ilosc_przelacznikow
    Widok.cpp, 30
    WIDOK.h, 34
instrukcja_obslugi
    Widok.cpp, 30
    WIDOK.h, 34

konfrontuj_pokolenie_ze_srodowiskiem
    Model.cpp, 8
    MODEL.h, 20
konwertuj_przelaczniki
    Model.cpp, 9
    MODEL.h, 21
krzyzuj_pokolenie
    Model.cpp, 9
    MODEL.h, 21

litera_w_pliku
    Widok.cpp, 30
    WIDOK.h, 35
losuj_pare
    Model.cpp, 10
    MODEL.h, 22

main
    Main.cpp, 5
Main.cpp, 5
    main, 5
Model.cpp, 5
    czy_double, 7
    czy_int, 7
    dopisz_enter, 8
    konfrontuj_pokolenie_ze_srodowiskiem, 8
    konwertuj_przelaczniki, 9
    krzyzuj_pokolenie, 9
    losuj_pare, 10
    modyfikuj_przez_pokolenia, 11
    naprawa_strumienia, 11
    pobierz_koncowke, 12
    scal_chromosomy, 12
    sprawdz_dopasowanie, 13
    sprawdz_koniec, 13
    tworz_nowe_pokolenie, 14
    usun_skopiowane, 14
    wczytaj_z_pliku, 15
    wypisz_do_pliku, 15
    zamien_z_kopia, 16
    znajdz_krzyzowany, 16
    znajdz_najwieksza_srednia, 17
MODEL.h, 17
    czy_double, 18
    czy_int, 19
    dopisz_enter, 20
    konfrontuj_pokolenie_ze_srodowiskiem, 20
    konwertuj_przelaczniki, 21
    krzyzuj_pokolenie, 21
    losuj_pare, 22
    modyfikuj_przez_pokolenia, 22
    naprawa_strumienia, 23
    pobierz_koncowke, 23
    scal_chromosomy, 24
    sprawdz_dopasowanie, 25
    sprawdz_koniec, 25
    tworz_nowe_pokolenie, 26
    usun_skopiowane, 26
    wczytaj_z_pliku, 27
    wypisz_do_pliku, 27
    zamien_z_kopia, 28
    znajdz_krzyzowany, 28
    znajdz_najwieksza_srednia, 29
modyfikuj_przez_pokolenia
    Model.cpp, 11
    MODEL.h, 22

naprawa_strumienia
    Model.cpp, 11
    MODEL.h, 23

plik_wejsciowy.txt, 29
plik_wyjsciowy.txt, 29
pobierz_koncowke
    Model.cpp, 12
    MODEL.h, 23
pobierz_przelacznik
    Widok.cpp, 31
    WIDOK.h, 35

README.md, 29

scal_chromosomy

Model.cpp, 12
MODEL.h, 24
sprawdz_dopasowanie
Model.cpp, 13
MODEL.h, 25
sprawdz_koniec
Model.cpp, 13
MODEL.h, 25
sprawdz_otwarcie
Widok.cpp, 31
WIDOK.h, 36
sprawdz_poprawnosc_przelacznikow
Widok.cpp, 33
WIDOK.h, 36

tworz_nowe_pokolenie
Model.cpp, 14
MODEL.h, 26

usun_skopiowane
Model.cpp, 14
MODEL.h, 26

wczytaj_z_pliku
Model.cpp, 15
MODEL.h, 27
Widok.cpp, 29
ilosc_przelacznikow, 30
instrukcja_obslugi, 30
litera_w_pliku, 30
pobierz_przelacznik, 31
sprawdz_otwarcie, 31
sprawdz_poprawnosc_przelacznikow, 33
wypisz_populacje, 33
WIDOK.h, 34
ilosc_przelacznikow, 34
instrukcja_obslugi, 34
litera_w_pliku, 35
pobierz_przelacznik, 35
sprawdz_otwarcie, 36
sprawdz_poprawnosc_przelacznikow, 36
wypisz_populacje, 37
wypisz_do_pliku
Model.cpp, 15
MODEL.h, 27
wypisz_populacje
Widok.cpp, 33
WIDOK.h, 37

zamien_z_kopia
Model.cpp, 16
MODEL.h, 28
znajdz_krzyzowany
Model.cpp, 16
MODEL.h, 28
znajdz_najwieksza_srednia
Model.cpp, 17
MODEL.h, 29