Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Darwin

autor	Szymon Ostrzołek
prowadzący	mgr inż. Anna Kuliś
rok akademicki	2018/2019
kierunek	Teleinformatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium / ćwiczeń	środa, 10:15 – 11:45
grupa	2
sekcja	4
termin oddania sprawozdania	2019-01-25
data oddania sprawozdania	2019-02-08

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 wylosowanych jest k par osobników, które się następnie krzyżują. Krzyżowanie polega na tym, że u każdego osobnika dochodzi do pęknięcia chromosomu w dowolnym miejscu. Część 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 $w <= f <= r$ pozostają w populacji, ale się nie rozmnażają.

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

Plik wejściowy ma następującą postać: Każda linia zawiera jednego osobnika. Osobnik charakteryzowany jest chromosomem, który jest przedstawiany jako ciąg liczb naturalnych rozdzielonych białymi znakami.

Przykładowy plik wejściowy zawierający populację złożoną 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 wyjściowy ma identyczny format.

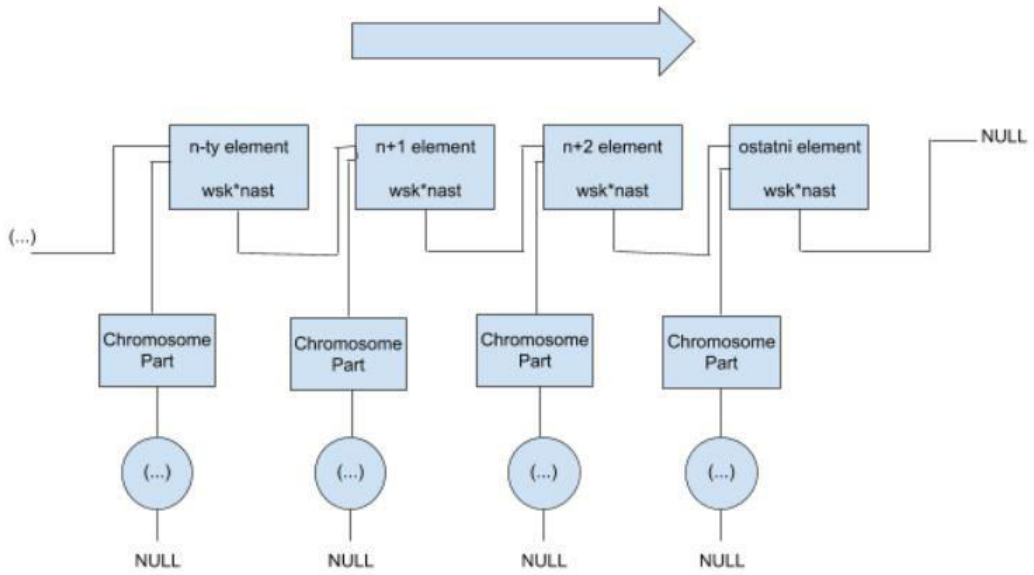
2 Analiza zadania

Zagadnienie przedstawia problematykę pobierania danych z pliku, dynamicznego ich modyfikowania oraz zapisywania wyniku tej modyfikacji z powrotem do pliku.

2.1 Struktury danych

W programie wykorzystałem listy oraz ich “odmianę”, a mianowicie listy podwieszane. Lista jednokierunkowa jest strukturą o dynamicznie zmieniającej się wielkości. Listę można opisać jako uszeregowany zbiór elementów. Każdy element zawiera jakieś dane oraz wskazuje na swojego następcę. Cechą listy jednokierunkowej jest to, że można przeglądać ją tylko w jedną stronę, od początku do końca. Każdy element listy ma podwieszone do siebie kolejne elementy (ilość tych elementów zależy w tym przypadku od długości chromosomu) tworząc tak zwaną listę podwieszaną.

2.2 Algorytmy



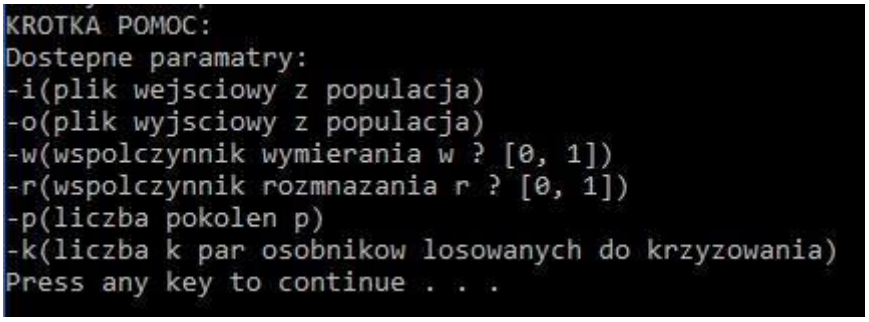
3 Specyfikacja zewnętrzna

Program uruchamiany jest z linii poleceń. Wpisujemy nazwę programu, po tym należy przekazać odpowiednie przełączniki w obojętnej kolejności:

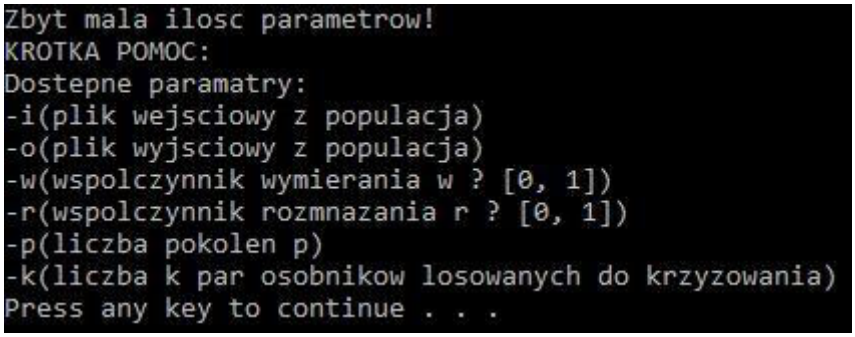
```
-i
-o
-w
-r
-p
-k
```

Na początku program sprawdza ile flag podaliśmy uruchamiając program. Jeśli nie podano wszystkich, awięc mniej niż siedem program wyświetla komunikat o

zbyt małej ilości parametrów oraz funkcję help() po czym kończy pracę programu. W przeciwnym razie wykonuje się pętla for, tyle razy ile flag by za każdym przebiegu albo odczytać wartość podanej flagi albo wykonać funkcję podaną w odpowiadającym if. Podanie parametru -h wywołuje funkcję h(); zawierającą krótką pomoc.



Dodatkowo wypisanie błędnego parametru lub gdy brakuje parametru wyświetla komunikat o błędzie wraz funkcją h();



Przykładowo prawidłowym uruchomieniem programu będzie polecenie:

```
darwin.exe -i "”population.txt” -o “result.txt” -w -r -p 2 -k
```

-i “population.txt” przypisuje do zmiennej inputFilename nazwę pliku,
-o “results.txt” przypisuje do zmiennej outputFilename nazwę pliku,
-w uruchamia funkcję losującą współczynnik wymierania,
-r uruchamia funkcję losującą współczynnik rozmnażania,
-p 2 podaje, że w programie występują dwa pokolenia,
-k odpowiada za ilość par wylosowanych do krzyżowania

Na samym początku wykonuje się funkcja `in();` której zadaniem jest odczytanie zawartości pliku `“population.exe”` przez polecenie:

```
fstream handler;
handler.open(inputFilename, ios::in);
```

oraz zapisanie jego zawartości do listy podwieszanej. Wykonujemy to tworząc dwa `struct'y`. Pierwszy o nazwie `individual` tworzy listę osobników a drugi o nazwie `chromosomePart` jest "podczepiony" pod niego przyjmując kolejno każdą liczbę/cyfrę w danej linii, które oddzielone są białymi znakami lub spacjami. By to osiągnąć korzystamy z metody `getline`, która pobiera dane właśnie po linijce co jest dla nas idealnie pasującym sposobem. Tak więc każdy `getline` to pojedyńczy osobnik składający się z tylu `chromosomePart'ów` ile elementów w linii. Funkcja ta wykonuje się tyle razy ile istnieje pokoleń.
Następnie wykonuje się funkcja `crossPopulation();`. W tej funkcji obliczamy ile istnieje osobników by móc wylosować ilość par do krzyżowania czyli liczbę z zakresu od 0 do `howMuch`.
Teraz dla każdego osobnika wylosowanego do krzyżowania wysyłamy w pętli za pomocą wskaźnika elementy chromosomu od elementu po złamaniu aż do końca chromosomu do drugiego osobnika z pary i to samo robimy w drugą stronę.
Następnie dla każdego osobnika wywołujemy funkcję `randR();`, `randW();` oraz `randD();` i wypisujemy je na ekranie. Następnie instrukcją warunkową porównujemy wartość funkcji `randD();` z pozostałymi i podejmujemy odpowiednie działanie:
-klonowania (wysłania osobnika podwójnie)
-usuwania (czyli po prostu pozostawienie bez przesłania)
-przesyłania osobnika do funkcji `out();`.

W funkcji out(); działamy bardzo podobnie jak w funkcji in(); lecz używamy innego parametru std::ios:

```
fstream handler;
handler.open(outputFilename, ios::trunc | ios::out);
```

W funkcji `out();` przesyłamy strumieniowo każdego osobnika do pliku, którego nazwa jest zapisana w zmiennej outputFilename.

5 Testowanie

Program został przetestowany na odporność na białe znaki oraz inne znaki. Program odporny jest na błędne pliki, gdy plik jest pusty lub nie istnieje wyświetla się stosowna informacja. Jest również odporny na błędne parametry podawane przy uruchamianiu programu oraz na braki w parametrach.

6 Wnioski

Program do modyfikacji chromosomów osobników w populacji był dość złożony. Najcięższym zadaniem było samo krzyżowanie osobników z racji konieczności wymieniania elementów listy podwieszanej z czym ostatecznie nie dałem sobie rady. Prócz tego udało mi się zapisywać do owej listy podwieszanej zawartość pliku zawnętrznego txt oraz wysyłanie jej do innego pliku z wynikiem krzyżowania. Była to najbardziej czasochłonna część całego projektu.