[Date]

ross erskine

[Company name]

[Company address]

Genetic algorithm

Specification documents

## Data dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class:**  **Person** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| mChromosome | String | Private member,  Only: ‘0’,’1’, of specific length | 0110,1100,1010 | represents byte and a person chromosome |
| mFitness | Double | Private member, within bounds of formula | 54, 1045 | represents a person’s fitness usually created using formula representing line on graph i.e. linear, curved |
| mDecodedInt | Int | Private member, > 0, < bounds of size of chromosome | 14,254,1023 | the value the chromosome represents used for arithmetic of fitness and fitnessRatio |
| mFitnessRatio | Double | Private member, ∑ of population’s fitnessRatio = 100 | 25,10, 4.2 | for use of selection functions |
| mCoordinates | Point[] | Private member, vector size = chromosome\_size / 2 | 6,8,12,16 | Represents coordinates on GUI. each chromosome (11,01)represents the next move on GUI. 00 = stay still, 01 = left, 10 = right, 11 = move forward. A Person with chromosome that equaled termination would move directly to target square. |
| **Function** | **Return type** | **Parameters** | **Description** | |
| createCoordinates | Void | n/a | Creates point coordinates from chromosomes | |
| Is\_chromosome | Bool | string | Makes sure that chromosome is either (1,0) | |

ConstVariable represents global variables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class:**  **ConstVariables** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| mChromosome\_size | Int | Constant, private member | 6,8,12,16 | Represents the number of chromosomes a Person has. |
| mPopulation\_size | int | Constant, private member | 6,8,10,12 | Represents the number of populations in each generation |
| mTermination | Double | Constant, private member | 255,655.35 | Represents the maximum number a chromosome can reach. |
| mCrossover\_rate | Double | Constant, private member | 0.7 | Represents the probability of crossover, otherwise *cloning* takes place |
| mMutation\_rate | Double | Constant, private member | 0.001 | Represents the probability of mutation to take place |
| mGaDataFile | String | Constant, private member | “gaDataFile” | Start of filename |
| mDotcsv | String | Constant, private member | “.csv” | End of filename |
| mInstructions | Int | Constant | 50, 1000, 10000 | Desired number of generations |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Functions:**  **Population** |  |  |  |  |
| **Function** | **Return type** | **Parameters** | **description** | |
| GeneratePopulation | Void | Person[], constant int | Creates chromosome using char characters to desired size, decode\_int works the value the chromosome number represents, then generates fitness from that value. after the whole population is completed the fitness ratio is worked out against the sum of all fitness | |
| Decode\_int | Void | Person | generates an int from the representation from chromosome i.e., 0101 = 5 | |
| generateFitness | Void | Person | generates fitness from decoded int 'x' based on formula where 'T' is highest number possible, so termination value is used | |
| populationAvgFitness | Double | Constant Person[] | Works out average fitness on every generation used to compare generations on graph based on formula , where is average population fitness, p is population fitness, is sum of all fitness and is population size | |
| Fitness\_ratio | Double | Constant Person[], Person | Fitness ratio is worked out after population is filled based on formula where is personal fitness is the sum of all fitness and is ratio | |
| Print\_population | Void | Constant Person[] | prints population on console mainly for testing and debugging | |
| Is\_termination | Bool | Constant Person[] | Looks for a chromosome in population equal to termination, termination usually maximum chromosome can be | |
| highestFitness | Double | Constant Person[] | Returns highest fitness from a population from each generation for comparison of generations used in graphs | |
| bubbleSort | Void | Person[] | sorts population based on fitness ratio | |
| Roulette\_wheel | Void | Person[], Person, Person | Roulette wheel sorts the population by fitness ratio using bubblesort selects two random numbers for mum and dad from 1 to 100. selection fitness ratio stacks proportional sections depending on size of ratio if the random numbers are in a Persons section a Person is picked | |
| Tournament | Void | Person[], Person, Person | Picks top 4 Persons based on fitness ratio puts them a semi-finalist container then randomly selects two from that pot mum and dad cannot be same | |
| rankSelection | Void | Person[], Person, Person | ranks persons in order of fitness, then selection is random but, probability is equal example: population of 10 would be probability of 0.1 each | |
| Crossover | Void | Person[], Person, Person | Splits parent chromosome up to create child chromosome only splits with a probability of 0.7 known as crossover rate or just replace child with copy of mum and dad any split is random generated by a number between 0 and 1 .all children are handed to mutation function | |
| Mutation | Void | Person | Only mutates child with a probability represented as random number generated between 0.001 and 1. | |
| rePopulate | Void | Person[] | rePopulate generates numbers for population and repopulates remaining population after new child is added and decoded int and fitness added to child rest of population is then regenerated to get population back to size | |
| Is\_n\_population | Bool | Constant Person[] | Checks size of population | |
| getGA | Void | Int, Constant Person[] | Creates initial population and loops through instructions of generations or finds termination criteria. main purpose so draw loop creates the loop of generations | |
| gaOutData | Void | Const, int, const long long int, const double, const string, const bool, const string | csv file upload with data: generation, average fitness, highest fitness and if termination has been reached for each generation | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class:**  **GA\_window** |  |  |  |  |
| **Field** | **Datatype** | **Validation** | **Example** | **description** |
| mGeneration | Int | Private member, >0 | 1,50,100,2000 | The number of generations normally loops |
| mChromosomeCount | Int | Private member, 0 >, < = 8 | 1,2,8 | Helps game\_loop() count number iterations redraws before a new generation is needed |
| mHorde | MyCircle[] | Private member, | 6 | Places circles into a set that can have its coordinates redrawn during a generation or many generations |
| currPopulation | Person[] | Private member | 12 | Ga\_window’s representation of population |
| **Function** | **Return type** | **Parameters** | **description** | |
| Cb\_game\_loop | Static void | Address | Call-back handle from fltk library | |
| Game\_loop | Void |  | Uses redraw to move Horde[] circles | |

## Tests

## Algorithm tests

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| Test id | Fitness type | Chromosome size | Population size | Termination criteria | Crossover rate | Mutation rate | Selection | Instructions | Result |
| 07 |  | 16 | 12 | 655.35 | 0.7 | 0.001 | Rank selection | 2000 |  |
| 08 |  | 16 | 12 | 655.35 | 0.7 | 0.01 | Rank selection | 2000 |  |
| 09 |  | 16 | 12 |  | 0.7 | 0.1 | Rank selection | 2000 |  |
| 10 |  | 16 | 12 |  | 0.3 | 0.001 | Rank selection | 2000 |  |
| 11 |  | 16 | 12 |  | 0.7 | 0.001 | Tournament | 2000 |  |
| 12 |  | 16 | 12 |  | 0.7 | 0.01 | Tournament | 2000 |  |
| 13 |  | 16 | 12 |  | 0.7 | 0.1 | Tournament | 2000 |  |
| 14 |  | 16 | 12 |  | 0.3 | 0.001 | Tournament | 2000 |  |
| 15 |  | 16 | 12 |  | 0.7 | 0.001 | Roulette wheel | 2000 |  |
| 16 |  | 16 | 6 |  |  |  |  |  |  |
| 17 |  | 8 | 6 |  | 0.7 | 0.001 | Roulette wheel | 2000 |  |
| 18 |  | 16 | 12 |  | 0.7 | 0.1 | Roulette wheel | 2000 |  |
| 19 |  | 16 | 12 |  | 0.3 | 0.001 | Roulette wheel | 2000 |  |

# 

# Tests

### Key

|  |  |
| --- | --- |
| Test type | Colour Code |
| Valid |  |
| Valid Boundary |  |
| Invalid |  |
| Invalid Boundary |  |
| Erroneous |  |

\*Error handled by programmer rather that compiler.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class: Person | Operation: class members | | | | | |
| Attribute | ID | | Test Data | Expected Results | Actual Results | Notes |
| setChromosome/  getChrromosome  (on 4 chromosome size) | 1.1.0 | 1.1.1 | 0101 | Set data |  |  |
| 1.1.2 | 1111 | Set data |  |  |
| 1.1.3 | 0101 | \*Error handled |  |  |
| 1.1.4 | 0102 | Error handled |  |  |
| 1.1.5 | dave | Error handled |  |  |
| setFitness/  getFitness | 1.2.0 | 1.2.1 | 54.2 | Set data |  |  |
| 1.2.2 | 0 | Set data |  |  |
| 1.2.3 | -1 | Error handled |  |  |
| 1.2.4 | 3000 | Error handled |  |  |
| 1.2.5 | \* | Error handled |  |  |
| setDecodedInt/  getDecodedint | 1.3.0 | 1.3.1 | 5 | Set data |  |  |
| 1.3.2 | 15 | Set data |  |  |
| 1.3.3 | 20 | Error handled |  |  |
| 1.3.4 | -1 | Error handled |  |  |
| 1.3.5 | 2.1 | Error handled |  |  |
| setFitnessRatio  getFitnessRatio | 1.2.0 | 1.2.1 | 22.2 | Set data |  |  |
| 1.2.2 | 0 | Set data |  |  |
| 1.2.3 | -1 | Error handled |  |  |
| 1.2.4 | 3000 | Error handled |  |  |
| 1.2.5 | \* | Error handled |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| interface: Population | Operation: Constructors | | | | | |
| Attribute | ID | | Test Data | Expected Results | Actual Results | Notes |
|  | 2.1.0 | 1.1.1 |  |  |  |  |
| 1.1.2 |  |  |  |  |
| 1.1.3 |  |  |  |  |
| 1.1.4 |  |  |  |  |
| 1.1.5 |  |  |  |  |
|  | 2.2.0 | 1.2.1 |  |  |  |  |
| 1.2.2 |  |  |  |  |
| 1.2.3 |  |  |  |  |
| 1.2.4 |  |  |  |  |
| 1.2.5 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Class: | Function: | | | | | |
| Attribute | ID | | Test Data | Expected Results | Actual Results | Notes |
|  | 1.1.0 | 1.1.1 |  |  |  |  |
| 1.1.2 |  |  |  |  |
| 1.1.3 |  |  |  |  |
| 1.1.4 |  |  |  |  |
| 1.1.5 |  |  |  |  |
|  | 1.2.0 | 1.2.1 |  |  |  |  |
| 1.2.2 |  |  |  |  |
| 1.2.3 |  |  |  |  |
| 1.2.4 |  |  |  |  |
| 1.2.5 |  |  |  |  |