|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data Structures and Subroutines | |  |  |  | | | |
| **Simulator.cs – Subroutines** | |  |  |  | | | |
| **Identifier** | **Return Type** | | **Parameters** | **Description** | | | |
| **Input Tabs**  These subroutines relate to the text editor contained by the InputWindow control. This includes:   * Creating and deleting machines; * Saving and opening machines to and from text files; * Renaming machines; * Duplicating machines; * Controlling what content is pasted into machines. | | | | |  |  |  |
| Tab\_AddTab | void | | object sender, EventArgs e | Adds a new machine tab to the inputWindow object when the ‘New Machine’ button is pressed or the shortcut is activated. | | | |
| Tab\_AddTab | void | | string Text, int Machine, int Tapes, List<int> point, List<string> Tape, List<string> State, string filename | Adds a new machine when the data is imported from a text file. | | | |
| Tab\_AddTab | void | | string Text, MachineData data | Adds a new machine when the data is duplicated from an existing machine. | | | |
| Tab\_RemoveTab | void | | object sender, EventArgs e | Deletes a machine when the ‘Delete Machine’ button is pressed. | | | |
| Tab\_SaveTab | void | | object sender, EventArgs e | Saves machine data to a text file in a format that allows it to be re-opened by the machine. | | | |
| Tab\_OpenTab | void | | object sender, EventArgs e  OR  string path | Opens a machine from a text file in the correct format.  OR  Opens a machine from the Debug file with the filename specified by the parameter. Used to open example files. | | | |
| Tab\_OpenExample | void | | object sender, EventArgs e | Loads the example machine chosen by the user in the menu. | | | |
| Tab\_Duplicate | void | | object sender, EventArgs e | Creates a copy of the currently selected machine. | | | |
| Tab\_Rename | void | | object sender, EventArgs e | Opens a window that allows the user to enter a new name for the selected tab. If this is not a blank string, it will be accepted and rename the machine. | | | |
| Tab\_Paste | void | | object sender, KeyEventArgs e | Removes any media from the clipboard that is not text before pasting to the text editor. | | | |
| **Set UI From Tags During Creation**  The subroutines in this section are used to set the initial values of additional elements in the UI. This includes:   * Initially setting what UI elements can be seen by the user (based on machine variant); * Adding the correct number of tapes to the program; * Configuring every tape’s tape, read/write head, and initial state controls. | | | | |  |  |  |
| Tag\_SetTape | void | | List<string> Tape, List<int> Pointers, List<string> States, int Machine | Sets the values of the added tapes after creating a new machine. | | | |
| Tag\_SetMachine | void | | int value | Sets the index of the machine drop-down box to the new machine value. Called when changing machines, converting machines and opening new machines. | | | |
| Tag\_SetTapes | void | | int Tapes | Adds new tape GUI elements when creating a new machine. | | | |
| **Update UI Post-Creation**  This section contains subroutines involved in the maintenance of the UI after the machine has been created. These subroutines perform:   * Updating what UI elements are visible based on machine variant; * Updating the speed of the machine’s operation; * Update the initial state drop-down box for tapes; * Update the text size of the tape and keep the read/write head in line with the selected character; * Update the machine’s tag’s initial state field; * Moving the read/write head’s position on relative to the tape; * Updating the height of the window. | | | | |  |  |  |
| UI\_UpdateMachine | void | | object sender, EventArgs e | Updates the machine value in currentMachineData. Will show and hide combinations of GUI elements depending on the new machine type. | | | |
| UI\_SetSpeed | void | | object sender, EventArgs e | Sets the new value for the speed at which the simulation runs when editing the slider position of the speed field. | | | |
| UI\_UpdateStates | void | | object sender, KeyEventArgs e  OR  ComboBox cb, string Text | When the transition function code is edited, the initial state drop-down boxes of each tape will be updated. If the existing current state is deleted, the selected state will become null.  OR  When changing machines, updates the initial state drop-down boxes of each state. | | | |
| UI\_UpdateTape | void | | object sender, EventArgs e | When the tape is edited by the user, this subroutine will ensure the information fits on the screen and then save the new tape to the MachineData of the current machine. Will also update the pointer’s position in relation to the tape by calling UpdatePointer(). | | | |
| UI\_UpdateInitialState | void | | object sender, KeyEventArgs e | When the initial state of any tape is changed, this subroutine will be called and will update initialStates list of the MachineData of the current machine to reflect this change. | | | |
| UI\_UpdatePointer | void | | Panel tape, int Pointer, int increment | This will identify the object that contains the pointer. Then, it will update | | | |
| UI\_PointerAdd | void | | object sender, EventArgs e | Runs the UpdatePointer() method with an increment of 1 when the right pointer button is pressed on a tape. | | | |
| UI\_PointerSubtract | void | | object sender, EventArgs e | Runs the UpdatePointer() method with an increment of -1 when the left pointer button is pressed on a tape. | | | |
| SetHeight | void | | int Tapes | Changes the height of the program when a new tape is added to make room for it, and when a tape is removed. | | | |
| ChangeMachine | void | | object sender, EventArgs e | Copies the data from the new MachineData object attached to the new tab page into the controls of the form. | | | |
| **Tape Instancing**  These subroutines are used to create and remove instances of tape objects from the scene.  Removing a tape can be done destructively or non-destructively;  Destructively removing a tape also removes the stored values of the tape from the machine – which is used when closing a tape manually.  Non-destructively removing a tape removes the tape’s UI object, but keeps its information. This is used when switching tapes, as the tapes in the scene are all removed and then added back in.  Creating tapes can also be done two ways;  When manually adding in a new tape, a new set of blank values are added to the machine’s storage.  When adding in only a new instance of a tape, no new values are added but existing values are used. This subroutine is used when changing machines and adding in the new tapes. | | | | |  |  |  |
| CloseTape | void | | Panel item | Temporarily removes the GUI tape elements while switching machines. | | | |
| CloseTape | void | | object sender, EventArgs e | Permanently closes a tape, removing the GUI elements and removing the data from the currentMachineData object. | | | |
| AddTape | void | | none | Adds new tape GUI elements without adding new data to the currentMachineData object. Used when changing machines and adding new machines. | | | |
| AddTape | void | | object sender, EventArgs e | Adds a completely new tape, including spawning new GUI elements and adding a new set of values to the currentMachineData object. | | | |
| **Runtime**  This section contains the subroutines responsible for running the Turing machines. These include:   * Setting the initial values needed to run the machine; * Checking that the machine is valid and does not contain errors; * Pausing the machine during operation; * Updating the UI once the machine has halted; * Resetting the machine after this is chosen by the user; * Performing a single step of the program; * Performing multiple steps of the program. | | | | |  |  |  |
| StartMachine | void | | object sender, EventArgs e | This method disables all controls but the ‘Pause Machine’ button and certain display objects to prevent them from appearing greyed-out. The machine will then be compiled in the Machine static class. | | | |
| RunMachine | Void | | object sender, EventArgs e | Will reset the values of all the current state and step counter objects in the scene, then calling StartMachine to compile the machine. Enables the runTimer to begin execution of the machine. | | | |
| CheckMachine | bool | | none | This ensures all of the data in the machine is valid, such as each line having the correct format and the initial state field being valid. | | | |
| PauseMachine | void | | object sender, EventArgs e | Enables the run, reset and step buttons on the GUI, and pauses the machine. | | | |
| StopMachine | Void | | object sender, EventArgs e | Enables all controls, except from the pause and reset buttons, and resets the tape(s)’ contents and pointer. | | | |
| HaltMachine | Void | | object sender, EventArgs e | Enables the ability to reset the machine once it has halted. | | | |
| StepMachine | Void | | object sender, EventArgs e | Performs one cycle of the Turing machine. Will be called every time the runTimer object ticks. | | | |
| FastStepMachine | void | | object sender, EventArgs e | Performs a large number (default 50) of steps of the Turing machine. Will be called every time the fastTimer object ticks, when the “Run at Full Speed” option is checked. | | | |
| StepStartMachine | void | | object sender, EventArgs e | Compiles the machine if not done so already, and then runs the StepMachine subroutine once without enabling the runTimer, to prevent concurrent steps. | | | |
| **Utility**  These subroutines are:   * Blocks of code that are reused a number of times and placed here to reduce repetition; * Methods that perform miscellaneous tasks that do not fit into any of the previous categories; | | | | |  |  |  |
| DisableControls | void | | Control C | Disables all controls that has C as its parent at some point in the hierarchy. Also disables C. | | | |
| EnableControls | void | | Control C | Enables control C and all controls it is parented to recursively. | | | |
| EnableAllControls | void | | Control C | Enables all controls contained by C recursively. | | | |
| ClearBox | void | | object sender, EventArgs e | Called when pressing the ‘Clear Debug Box’ button. Removes the contents of the ‘debug’ box, making it blank. Allows user to reduce visual clutter. | | | |
| ToggleSpeed | void | | object sender, EventArgs e | Called when changing the checked status of ‘isFullSpeed’. When turning this feature on, the slider to determine speed and the label displaying it are disabled and greyed out. When turning this feature off, they are enabled again. | | | |
| FillPointerLine | string | | int Pointer, int TapeLength | Creates a string that lines up the pointer with the tape. | | | |
| ConvertMachine | void | | object sender, EventArgs e | Takes a Deterministic Turing machine and generates a tape that is valid to use for the Universal Turing machine. | | | |
| LoadSettings | void | | none | Opens the “config.txt” file in the directory and loads its information into the program. | | | |
| QuitForm | void | | object sender, EventArgs e | Closes the form. | | | |
| **Open Forms**  The subroutines in this section are used to create new windows in the program that offer additional functionality. | | | | |  |  |  |
| CreateDiagram | void | | object sender, EventArgs e | Opens a GraphicalRepresentation form. | | | |
| OpenSettings | void | | object sender, EventArgs e | Opens the OptionsMenu form to allow the user to modify start-up preferences. | | | |
| OpenInfo | void | | object sender, EventArgs e | Opens the InfoWindow form to help the user learn how to use the program. | | | |

**Simulator.cs – Variables**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Data Type** | **Description** |
| TapeList | List<TapePanel> | Stores object references of each tape panel. |
| currentMachineData | MachineData | Stores the object reference of the current machine’s data, located in the current input window’s tab’s tag. |
| currentTab | TabObj | Stores an object reference for the currently selected tab page in the input window. |
| maxGraphicSize | const int | Constant value used when resizing a tape when its length is too large. |
| defaultCaseKey | char | The current character that is used in the machine as the default case character. |
| fullSpeedInterval | int | The number of steps the machine takes before updating the UI when the “Run at Full Speed” option is turned on. |
| sf | OpenFileDialog | An object reference used to open a navigation window to allow the user to select the save or open directory for their machine. |
| sr | StreamReader | An object used whenever reading data from a text file containing a Turing machine or a configuration file. |
| sw | StreamWriter | An object used to create a text file and write Turing machine data or configuration data to it. |

**Machines – Subroutines**

**Each machine works slightly differently from the others. Information for each machine has been included. Note that some machines work identically to the ‘standard’ Deterministic machine in some methods.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Parameters** | **Description** |
| Main | void | **Deterministic and Non-Deterministic:**  ref Tape Tape, List<StandardTransition> Commands, ref string CurrentState, ref bool IsHalted, string OverrideString  **Read-Only:**  ref Tape Tape, List<ReadOnlyTransition> Commands, ref string CurrentState, ref bool IsHalted, string OverrideString  **Multi-Tape and Universal:**  ref MultiTape Tape, List<MultiTapeTransition> Commands, ref string CurrentState, ref bool IsHalted, List<string> OverrideSequences  **Multi-Track:**  ref List<Tape> Tape, List<StandardTransition> Commands, ref List<string> CurrentStates, ref bool IsAllHalted, ref bool[] IsTapeHalted, string[] OverrideArray | **Deterministic, Read-Only, Multi-Tape and Universal:**  Called each tick during operation. Will check to find instructions that match the current state of the machine and call the step subroutine to validate them. Once a matching instruction is found, the method will stop checking functions. If no matching instruction is found, the machine will halt.  **Non-Deterministic:**  Instead of sorting through each transition function using iteration, this machine sends all of the functions to its Step procedure, where the matching functions are all generated in one call. This is because checking individual functions in Non-Deterministic machines is not possible as the definition for a read key is stretched across multiple functions.  **Multi-Track:**  Performs the same action as the Deterministic machine on each tape in the machine using iteration.  After completing this iteration, the program calls the IsAllHalted method to see if all of the tapes in the machine are halted. If so, it will return true through the IsAllHalted reference parameter. |
| Step | void | **Deterministic:**  StandardTransition code, ref Tape Tape, ref string CurrentState, ref bool isStepCompleted, string OverrideString  **Read-Only:**  ReadOnlyTransition code, ref Tape Tape, ref string CurrentState, ref Boolean IsStepCompleted, string OverrideString  **Non-Determninistic:**  List<StandardTransition> commands, ref Tape Tape, ref string CurrentState, ref bool IsStepCompleted, string OverrideString  **Multi-Tape and Universal:**  MultiTapeTransition code, ref MultiTape Tape, ref string CurrentState, ref bool IsStepCompleted, List<string> OverrideSequences | **Deterministic and Multi-Track:**  Checks to see if the given instruction is valid. If so, will call the MoveTape (and WriteTape) methods to update the tape.  **Read Only:**  Does not call the WriteTape method if the instruction is passed.  **Non-Deterministic:**  The program checks the whole set of transition functions for functions with the same state and read key. The program compiles a list of all matching functions and uses a random number generator to get a random matching function to execute.  **Multi-Tape and Universal:**  Checks that the given instruction is valid. If it is, the program goes through the list of override sequences to find the sequence of the highest priority that matches the tape. If this sequence is the same as the given instruction, this function will be executed. If not, the functions is rejected. |
| WriteTape | void | **Deterministic, Non-Deterministic and Multi-Track:**  char WriteKey, ref Tape Tape  **Read-Only:**  not used  **Multi-Tape and Universal:**  List<char> WriteKeys, ref MultiTape Tape | **Deterministic, Non-Deterministic and Multi-Track:**  Replaces the character at the current position with the new character specified by the instruction.  **Multi-Tape and Universal:**  For each tape in the machine, the currently ‘selected’ character on the tape is replaced with the new character specified by the write key of the instruction. |
| MoveTape | void | **Deterministic, Read-Only, Non-Deterministic and Multi-Track:**  char MoveKey, ref Tape Tape  **Multi-Tape and Universal:**  List<char> MoveKeys, ref MultiTape Tape | **Deterministic, Read-Only, Non-Deterministic and Multi-Track:**  Moves the read/write head of the tape in accordance with the given instruction.  **Multi-Tape and Universal:**  For each tape in the machine, the read/write head of the tape is moved according to the direction specified by the move key field of the instruction. |
| IsAllHaltedCheck | bool | **Only used in Multi-Track machine:**  bool[] IsHalted | Checks each element in the IsHalted array to see if all tapes on the machine are halted. If this is the case, true will be returned allowing the whole machine to halt. |

**UniversalMachine.cs – Subroutines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Parameters** | **Description** |
| GenerateUniversalCode | List<string> | string[] fields, char defaultCaseCharacter | Produces a list of transition functions to automatically produce code for a user to use in a Universal Turing machine. |
| ConvertMachine | void | ref MachineData currentMachineData, List<TapePanel> TapeList, ref RichTextBox code, char defaultCaseCharacter | Takes a Deterministic Turing machine and generates a tape that is valid to use for the Universal Turing machine. |

**Machine.cs – Subroutines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Parameters** | **Description** |
| isCompiled | Bool | none | Checks to see if the class contains initialised attributes, to ascertain if the machine is ready to be run or not. |
| isHalted | Bool | none | Checks if the machine is currently in a halted state. Will return true if it is, will return false if it is not. |
| Compile | Void | string Text, ref MachineData machineData, ref TextBox debug, ref List<TapePanel> Tapes | Transfers information from the UI to the class’ attributes in order to be able to run the machine. |
| Tick | Void | none | Performs one instruction in the operation of the machine. |
| Update | Void | ref List<TapePanel> Tapes, ref MachineData machineData, ref Label currentState, ref bool isHalted, ref bool[] isTapeHalted | Transfers information back to the UI after each tick operation to allow the user to view the changes after each instruction is performed. |
| DeCompile | Void | none | Sets the attributes that are given values to null in order to indicate to the program that the machine isn’t compiled. |
| GetOverrideChars | string | List<StandardTransition> Script  OR  List<ReadOnlyTransition> Script | **Used by Deterministic, Read-Only and Non-Deterministic:**  Produces a list of characters (as a string) of read keys that have transition functions defined for them in the current state of the machine. This list is used when deciding if a read-key should use the default case function or not. |
| GetOverrideArray | string[] | List<StandardTransition> Script, int tapeCount | **Used by Multi-Track:**  Produces a list of character lists (as strings) of read keys. Each string contains all of the characters that have been defined by a transition function of the current state of that tape (index of string corresponds to index of tape). These character lists are used when deciding if a tape in a Multi-Track machine should use the default case function. |
| GetOverrideSequences | List<string> | List<MultiTapeTransition> Script, int tapeCount | **Used by Multi-Tape and Universal:**  Produces a list of strings to be used in validating a Multi-Tape transition function.  The returned list contains every sequence of read keys that is defined in a transition function in the current state. These are used to determine what sequences can override the default case function. The list is also used to compare the currently read in function to determine if its priority is high enough to be executed (See Multi-Tape description for Step). |

**Machine.cs – Variables**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Data Type** | **Description** |
| machine | int | Used to identify which machine variant has been chosen by the user. |
| Tape | Tape | Stores the tape information for single tape Turing machines. |
| multiTape | MultiTape | Stores the tape information for Multi-Tape Turing machines. |
| multiTrack | List<Tape> | Stores the tape information for Multi-Track Turing machines. |
| currentState | string | Stores the current state of the machine for single tape and Multi-Tape Turing machines. |
| currentStates | List<string> | Stores the current states for Multi-Track Turing machines. |
| IsHalted | bool | Stores if the machine has been halted or not for all Turing machines. |
| IsTapeHalted | bool[] | Stores the operation state of each tape in a Multi-Track machine. |
| Commands | List<StandardTransition> | Used to store the commands to be used in a Deterministic, Non-Deterministic, Universal and Multi-Track Turing machine. |
| Commands1 | List<ReadOnlyTransiton> | Stores the commands to be used in a Read Only Turing machine. |
| Commands3 | List<MultiTapeTransition> | Stores the commands to be used in a Multi-Tape Turing machine. |
| OverrideString | string | Stores the characters that can override the default ‘\*’ key in the current state of the machine for single tape Turing machines. |
| OverrideArray | string[] | Stores the characters that can override the default ‘\*’ key in each current state of a Multi-Track Turing machine. |
| OverrideSequences | List<string> | Stores the character sets that can override the default case function for Multi-Tape and Universal Turing machines. |

**Graphical\_Representation.cs – Subroutines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Paramteters** | **Description** |
| Generate | void | object sender, EventArgs e | Called when the form opens. Controls the process of creating the diagram by calling the Spawn subroutine and producing a new EdgeCanvas object. |
| Spawn | void | string state, string parent | Recursive subroutine that initialises a node for each state in the graph in a depth-first method, with each node at an ascertained offset from their parent node. |
| RefreshEdges | void | object sender, EventArgs e | Whenever the form is scrolled, this method is called and forces the EdgeCanvas’ graphics object to refresh. This causes the edges on the object to be redrawn to be correct to the new relative positions of the state nodes on the screen. |

**Graphical\_Representation.cs – Variables**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Data Type** | **Description** |
| code | string | Stores the transition functions given by the main program on initialisation. Used to produce the graph. |
| initialState | string | Stores the initial state of the machine. This will be used to determine the root node in the graph. |
| graph | Graph | Stores two read-only adjacency lists for the machine, one keeping a list of the connecting nodes and the other, a corresponding list of the information about each transition. |
| spacing | const int | A constant to define the spacing between the rendered nodes. |

**EdgeCanvas.cs – Subroutines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Parameters** | **Description** |
| PaintEvent | void | object sender, PaintEventArgs e | Creates a graphics object and a pen to use for drawing on the canvas. Also determines the offset relative to the top-left hand corner of the control, which is needed when the form is scrolled in either axis. Calls when object is repainted or refreshed. |
| DrawEdge | void | Graphics g, Pen p, Graphical\_Representation parent, int xOffset, int yOffset | Draws every necessary edge on the canvas using the graph from the parent, which is given through the constructor. Three lines can be drawn; a straight, looping or arcing.  Straight lines are drawn when the final state is to the right of the initial state.  A looping line is drawn if the initial state is the same as the initial state, producing a Bezier curve that moves away from the centre of the graph and back to return to the node.  Arcing lines perform an arc around the outside of the graph to get to the final state. These are drawn when the conditions do not match any of the other two lines. |

**Graph.cs – Variables**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Data Type** | **Description** |
| graph | Dictionary<string, Dictionary<string, string>> | Stores a list of the transition functions for each state. The key is the starting state of the machine, and the value is a dictionary that stores all of the possible transitions the machine can perform in the key state. The key of the inside dictionary is the final state that the machine will find itself in, and the value of it is a string containing information about how to achieve this transition (read key) and what will happen during it (write and move keys). |
| neighbours | Dictionary<string, List<string>> | Stores a list of all of the states (halting or otherwise) adjacent to another. The key is the state the transition starts from and the list contains all of the states the machine can travel to from the key state. |
| statesHalting | List<string> | Stores a list of the states in the machine that do not have any transition functions coming out of it. This is needed to add the additional rings around these states. |
| isReadOnly | bool | Specifies whether the graph needs to represent a read-only machine or not. |

**Graph.cs – Subroutines**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Return Type** | **Parameters** | **Description** |
| ReassembleLine | string | string existingLine, string readKey, string writeKey, string moveKey, bool isReadOnly | When two transition functions come from the same state as each other and end in the same state as each other, this function produces a composite line which allows the two functions to be represented with only one edge in the graph. |
| GetReadKeys | List<string> | string line, ref int i | This returns a list of all of the read keys in a label for a transition function. It also returns the value of where the incrementer ‘i’ stops at so the GetWriteKeys() function can use this. |
| GetWriteKeys | List<string> | string line, ref int i | Returns a list of the write keys present in a label for a transition function. Also returns its value for ‘i’ to allow GetMoveKeys() to use it. |
| GetMoveKeys | List<string> | string line, ref int i | Returns a list of the move keys present in a label for a transition function. |
| AssembleLine | string | List<string> ReadKeys, List<string> WriteKeys, List<string> MoveKeys, bool isReadOnly | This takes a list of read, write and move keys and assembles them into a string of text that can be used as a label for a transition function. |