**Advanced Programming Report**

Project

|  |  |
| --- | --- |
| **Professor** | 심동규 교수님 |
| **Department** | Computer engineering |
| **Student ID** | 2014722030 |
| **Name** | 박현아 |
| **Class** | 설계 ( 1 ) / 실습 ( A ) |
| **Date** | 2015. 06.15 |

**Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

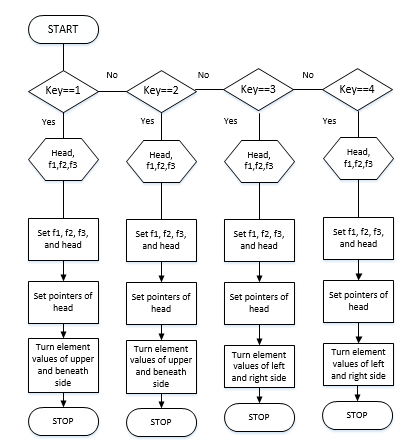
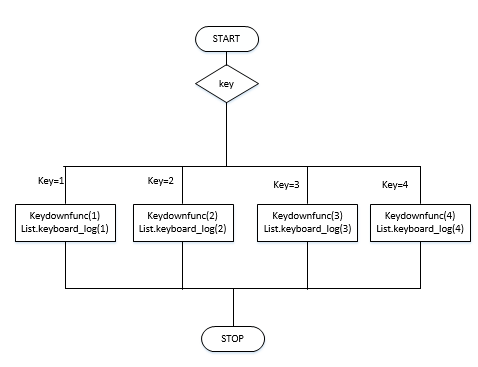
**[Introduction]**

Rubik’s cube is a puzzle that represents a large cube that consists of small cubes. Use MFC and print 3 face of cube. It can be changed according to input of mouse or keyboard. For example, if user input ‘↑’ of arrow keys, the front face becomes upside face. And then beneath face becomes front face. Note that elements of side faces also change. In operation according to what mouse clicks start point and end point, when start point and end point are input in program by using mouse, targeted slices make a rotation. And user can select whether print number on the screen or not. Finally, data that user input as using keyboard or mouse is stacked and when user click ‘keyboard log’ menu or ‘mouse log’ menu, each text file is generated and every data is stored in each text file.

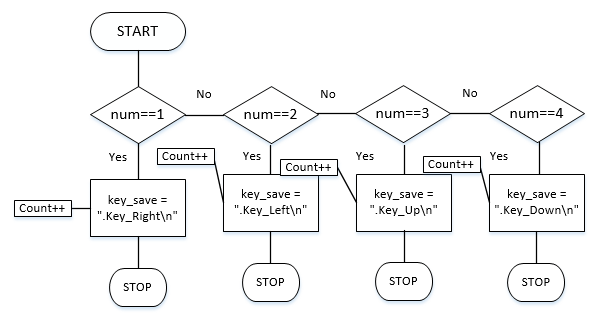
**[Flow Chart]**

<<Rotate a cube>>

1. OnKeyDown (2)keydownfunc

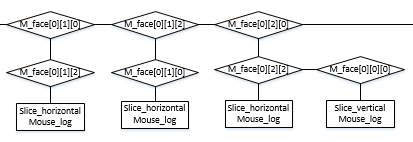
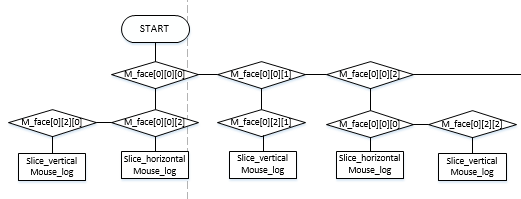


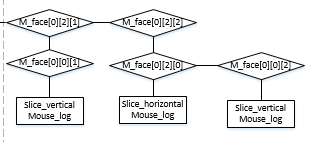
1. keyboard\_log



<<Rotate a slice of cube>>

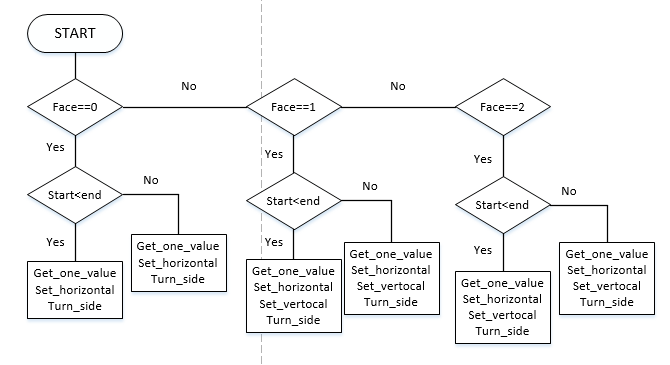
1. slice\_rotation



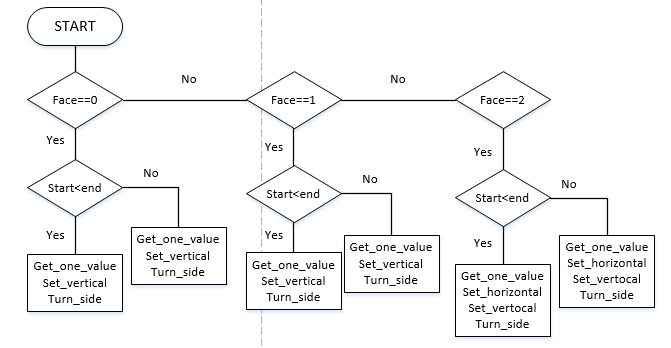


🡪Operation of m\_face[1] and m\_face[2] are same with operation of m\_face[0]. So I draw only operation of m\_face[0].

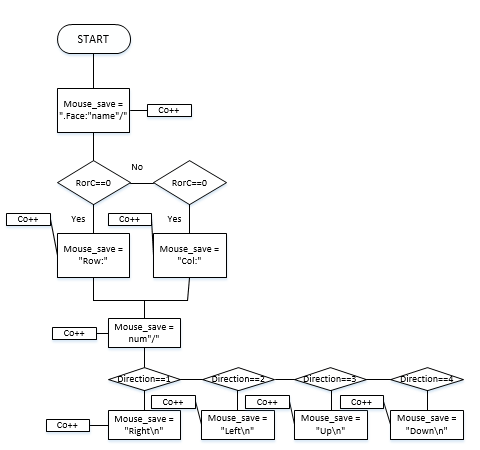
1. slice\_horizontal



1. slice\_vertical

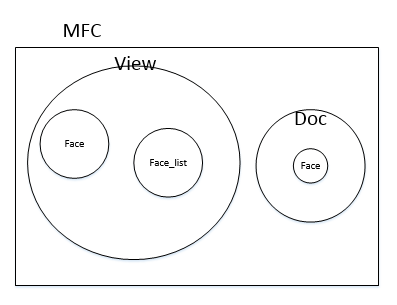


1. mouse\_log



**[Algorithm]**

<<Diagram of classes>



<<Face class>>

|  |  |
| --- | --- |
| **set\_name** | **char name** |
| This function sets a member variable ‘m\_face\_name’ to parameter ‘name’. And use this function when program initializes names of faces of a cube in constructor of ‘Face\_list’ class. | |
| **get\_name** |  |
| This function is enable to use member variable ‘m\_face\_name’ in other classes. | |
| **set\_value** | **int count** |
| This function sets a member variable array that is typed int ‘m\_element\_values’ to value of count and count increase when count becomes a value of one index of array one by one. And use this function when program initializes element values of faces of a cube in constructor of ‘Face\_list’ class. | |
| **get\_value** |  |
| This function is enable to use member variable ‘m\_element\_values’ in other classes. | |
| **set\_next** | **Face \* next** |
| This function sets a member variable pointer that points next node to make circular linked list to address of next node that parameter ‘next’ has. And use this function when program initializes element values of faces of a cube in constructor of ‘Face\_list’ class. Also use when rotate cube by using keyboard because, as the cube is rotated, the node of the linked list pointed to by header is changed. | |
| **get\_next** |  |
| This function is enable to use member variable ‘p\_Next’ in other classes. | |
| **set\_up** | **Face \* up** |
| This function sets a member variable pointer that points upper node to make circular inked list to address of upper node that parameter ‘up’ has. And use this function when program initializes element values of faces of a cube in constructor of ‘Face\_list’ class. Also use when rotate cube by using keyboard because, as the cube is rotated, the node of the linked list pointed to by header is changed. | |
| **get\_up** |  |
| This function is enable to use member variable ‘p\_Up’ in other classes. | |
| **get\_one\_value** | **int i, int j** |
| This function is similar with ‘get\_value’ function. The difference is that ‘get\_value’ function brings all of elements of a node but ‘get\_one\_value’ function brings only one value that row is ‘i’ and column is ‘j’. | |
| **set\_horizontal** | **int i, int \* temp** |
| This function is similar with ‘set\_value’ function. The difference is that ‘set\_value’ function sets all of elements by using parameter ‘count’ by increasing value of count one by one but ‘set\_horizontal’ function sets only elements that row is ‘i’ to values that temp has. | |
| **set\_vertical** | **int i, int \* temp** |
| This function is more similar with ‘set\_horizontal’ function than ‘set\_value’ function. The difference is that ‘set\_horizontal’ function sets elements that row is ‘i’ to values that temp has but ‘set\_vertical’ function sets elements that column is ‘i’ to values that temp has. | |
| **turn\_side** | **int num** |
| This function is to rotate elements 90 degrees clock wise or anti-clock wise. Parameter ‘num’ determine whether rotate clock wise or not. | |

<<Face\_list class>>

|  |  |
| --- | --- |
| **set\_head** | **Face \* head** |
| This function sets a member variable pointer ‘Face\_Header’ to parameter ‘head’. When program needs to change ‘Face\_Header’ in outer classes, change as using this function. | |
| **get\_head** |  |
| This function is enable to get ‘Face\_Header’ to outer classes. | |
| **slice\_horizontal** | **int face, int height, int start, int end** |
| This function operate events taken by mouse and work when row of start point end end point is same. There are four parameters. Parameter ‘face’ indicates whether an event did occur in some ways. And parameter ‘height’ means row. Finally parameter ‘start’ and ‘end’ indicate columns of start point and end point. As using these parameters, classify events according to condition. And use ‘get\_one\_value’, ‘turn\_side’, ‘set\_horizontal’, and ‘set\_vertical’ functions of ‘Face’ class. | |
| **slice\_vertical** | **int face, int width, int start, int end** |
| This function is similar with ‘slice\_horizontal’ function. The difference is that ‘slice\_horizontal’ function work when row of start point and end point is same but ‘slice\_vertical’ function work when column of start point and end point is same. ‘slice\_vertial’ function also has four parameters. Parameter ‘face’ indicates whether an event did occur in some ways. And width means column. Finally, parameter ‘start’ and ‘end’ indicate rows of start point and end point. As using these parameters, classify events according to condition, And use ‘get\_one\_value’, ‘turn\_side’, ‘set\_horizontal’, and ‘set\_vertical’ fucntions of ‘Face’ class. | |
| **keyboard\_log** | **int num** |
| This function stacks all of data input by keyboard. Parameter ‘num’ means each allow key. | |
| **key\_File\_out** |  |
| This function is called when user click ‘keyboard log’ menu. And then all of data that is stacked in ‘keyboard\_log’ function is send to a text file named “keyboard\_log”. | |
| **mouse\_log** | **char name, int RorC, char num, int direction** |
| This function stacks all of data input by mouse. There are four parameters. Parameter ‘name’ indicates face name that event did occur. And parameter ‘RorC’ means whether slice rotates vertical or horizontal. If ‘RorC’ is 0, it means horizontal. If ‘RorC’ is 1, it means vertical. And parameter ‘num’ means order of row or column. Finally, direction indicates the slice rotate toward. | |
| **mouse\_File\_out** |  |
| This function is called when user click ‘mouse log’ menu. And then all of data that is stacked in ‘mouse\_log’ function is send to a text file named “mouse\_log”. | |

<<View class>>

|  |  |
| --- | --- |
| **color\_mapping** | **int value, int \* brush** |
| This function saves color number in member variable array ‘brush’ according to value. | |
| **keydownfunc** | **int key** |
| This function works when events are taken place by keyboard and rolls a cube according to parameter ‘key’. | |
| **slice\_rotation** | **CPoint start, CPoint end** |
| This function works when events are taken place by mouse and turns the slice according to parameter ‘start’ and ‘end’. | |
| **OnDraw** | **CDC \* pDC** |
| This function draws a cube on the screen. | |
| **OnLButtonDown** | **UINT nFlags, CPoint point** |
| This function works when mouse clicks a place of the cube. And save a coordinate to member variable ‘start\_point’. | |
| **OnLButtonUp** | **UINT nFlags, CPoint point** |
| This function works when mouse button is up. And save a coordinate to member variable ‘end\_point’. | |
| **OnkeyDown** | **UINT nChar, UINT nRepCnt, UINT nFlags** |
| This function works when user push arrow keys on keyboard. According to keys, ‘keydownfunc’ and ‘keyboard\_log’ function of l ‘Face\_list’ class is called. | |
| **PopNumber** |  |
| This function works when user click ‘Color Number’ menu. And then initialize member variable ‘ID\_Menu\_Flag’ to TRUE. And call ‘OnDraw’ function. | |
| **OnlyColor** |  |
| This function works when user click ‘Only Color’ menu. And then initialize member variable ‘ID\_Menu\_Flag’ to FALSE. And call ‘OnDraw’ function. | |
| **KeyboarLog** |  |
| This function works when user click ‘keyboard log’ menu. And call ‘key\_File\_out’ function of ‘Face\_list’ class. | |
| **MouseLog** |  |
| This function works when user click ‘mouse log’ menu. And call ‘mouse\_File\_out’ function of ‘Face\_list’ class. | |

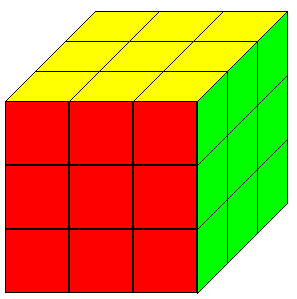
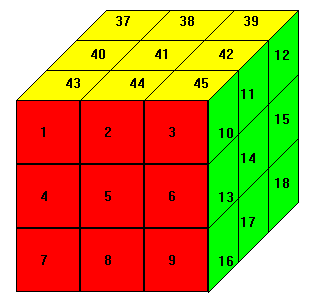
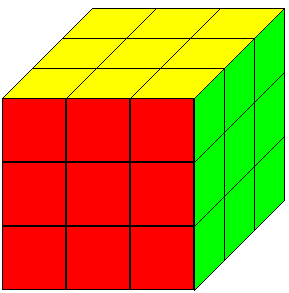
<<Doc class>>

|  |  |
| --- | --- |
| **set\_face** | **Face \* head** |
| This function initializes member variable ‘m\_face[3][3][3]’. ‘m\_face[0]’ has elements of parameter ‘head’. And ‘m\_face[1]’ has elements of upper face of ‘head’. Finally ‘m\_face[2]’ has elements of next face of ‘head’. | |

**[Execution result screen]**

<Select menu>

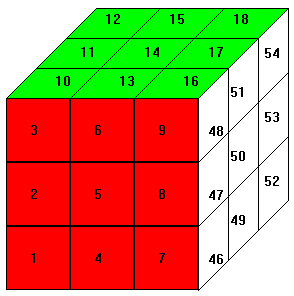
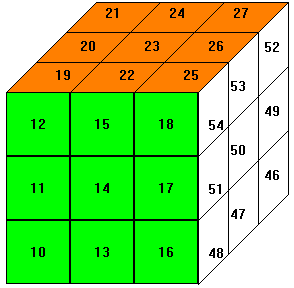
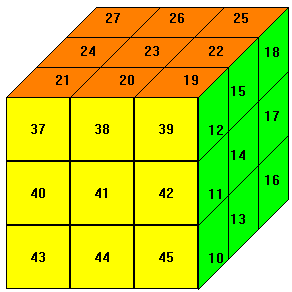
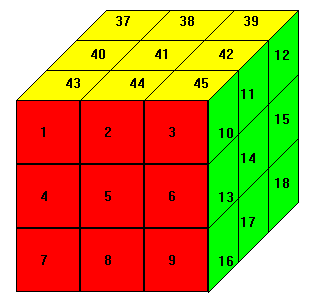
(1)Beginning screen -----------🡪(2) Select ‘Color Number’ menu------🡪(3) Select ‘Only Color’ menu

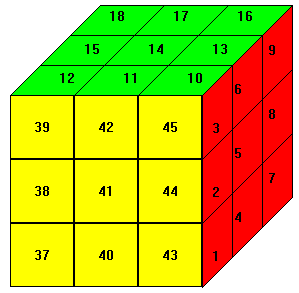
Beginning screen is set like (1)screen that doesn’t represent numbers on the screen. And when user select ‘Color Number’ menu, numbers represent on the screen. If user select ‘Only Color’ menu again, numbers become to disappear.

<Change screen by using KEYBOARD>

(1) Beginning screen----🡪(2) Operation of ’↓’-----🡪(3) Operation of ‘←’--🡪(4) Operation of ‘↑’-----

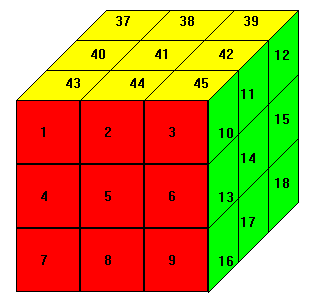
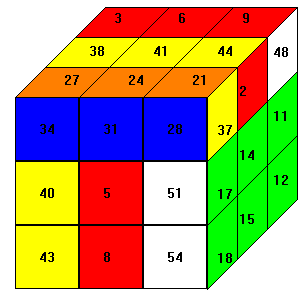
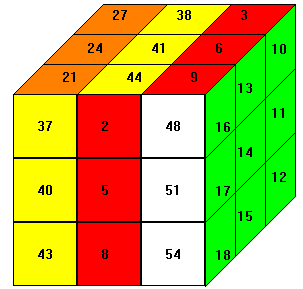


-----🡪(5) Operation of ‘→’

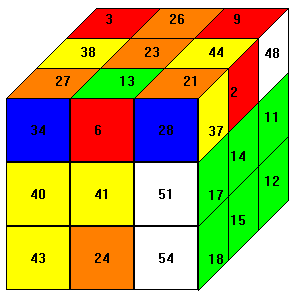
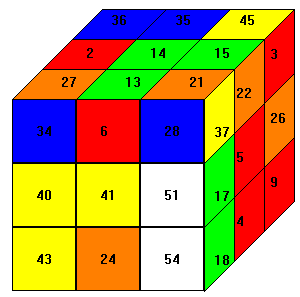
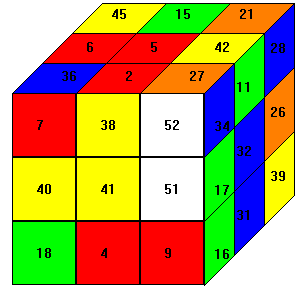
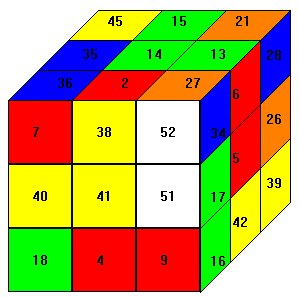
-If user push ‘↓’ button (if beginning screen is like to (1)), screen is changed such as (2). And then when user push ‘←’, ‘↑’, and ‘→’ one by one, screen is also changed correctly. Besides what pointers point is changed, elements of side faces or upper and beneath faces also change.

<Change screen by using MOUSE>

1. Beginning screen-----------------🡪(2)Operation of front face--------------------------------🡪

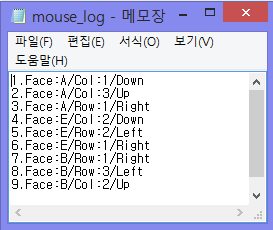
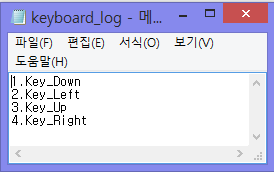
 

(3)Operation of upper face-----------------------------------🡪(4)Operation of next face

-First cube of (2) is to turn first column of front face down and to turn third column of front face up. And second cube of (2) is to turn first row of front face right. In operation of upper face, first cube of (3) is to turn second column down and second cube of (3) is to turn first row right and to turn second row left. Finally in peration of next face, first cube of (4) is to turn first and third row left. Second cube of (4) is to turn second column down.

<Save Text File>

Left capture screen is a result that I executed in <Change screen by using MOUSE> and right capture screen is a result that I executed in <Change screen by using KEYBOARD>

**[Consideration]**

When I make ‘color\_mapping’ function, I didn’t know how I can pass color number to ‘OnDraw’ function. First I tried to return CBrush variable. But this way doesn’t work. I don’t know the reason. So I tried second way. I made an array brush[3] in ‘OnDraw’ function. And then I gave brush as parameter. But the result was odd because value of first index sets correctly but the other values have garbage value. Once, I tired third way to allocate color. The third way is that I declare member variable array ‘brush[3]’ and ‘color\_mapping’ function doesn’t have return type. And I put ‘brush[3]’ as parameter of ‘color\_mapping’ function. It is possible because private member variable can be accessible in class that variable is declared.

When I make ‘keydownfunc’ function, I set head, next nodes, and upper nodes as using list.get\_head() directly. When I execute prgram early, It didn’t make any problem. But as time passes, there is a problem that two different nodes have same element values. I thought that some nodes have two pointers (p\_Up and p\_Next) and when program rotate the cube, relations become complexed. So I declare 3 pointers typed ‘Face’ and 3 pointers should indicate nodes that will change.

When I try to stack all of data that I input by using keyboard or mouse, I put integer value into char type array at first. But the result is odd because input integer values are regarded as ASCII table code number. So I try type conversion such as (char)number. But the result was same. I tried again. I changed the type of varialbe to ‘char’ and I initialize it ‘1’. It looks like that this code operate well. But when I input tenth data, 58th character of ASCII table represented. I agonized about this problem for a long time. As a result, I thought that number of order must not be stacked in array. So I just print order number in text file. To distinguish input data, I use ‘\n’. When program print data that is tacked in array one by one, if meets ‘\n’, program escape ‘for’ statement and print order number. And do this oepration again until program meets ‘\0’.