고급프로그래밍

2019203102
유지성

2020.06.20.

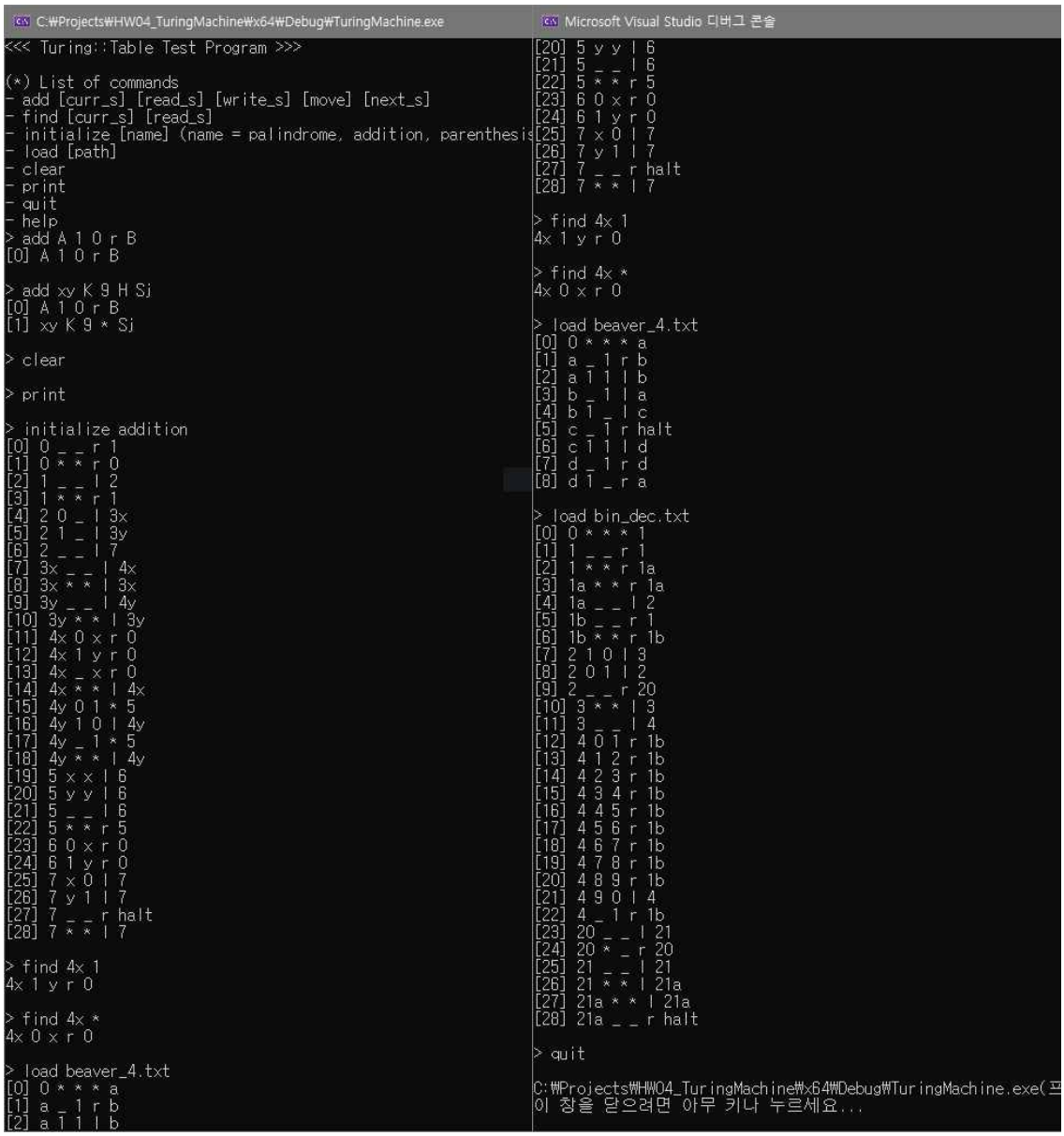
소프트웨어학부

요약

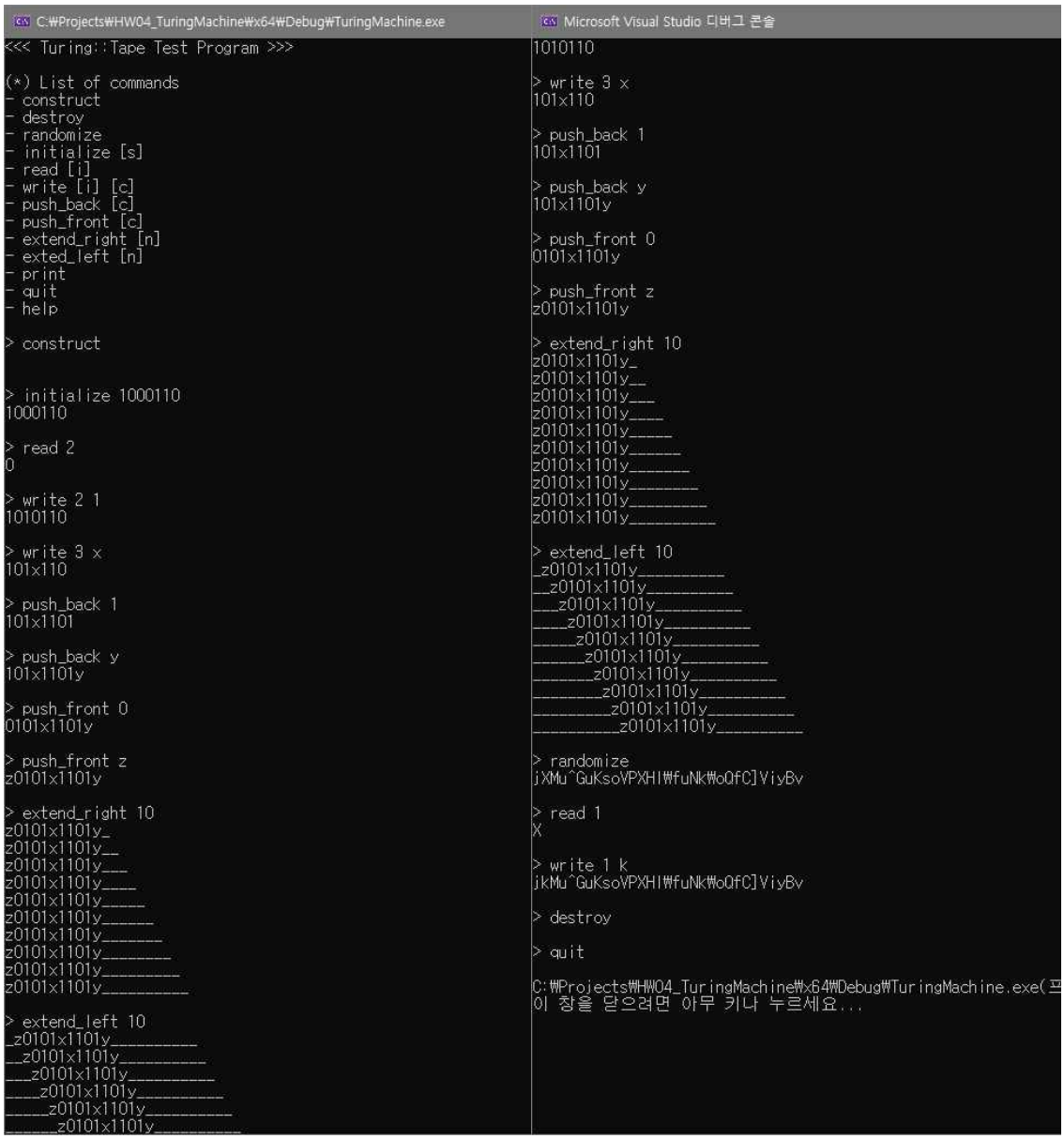
Table	O
Tape	O
Machine	O

결과

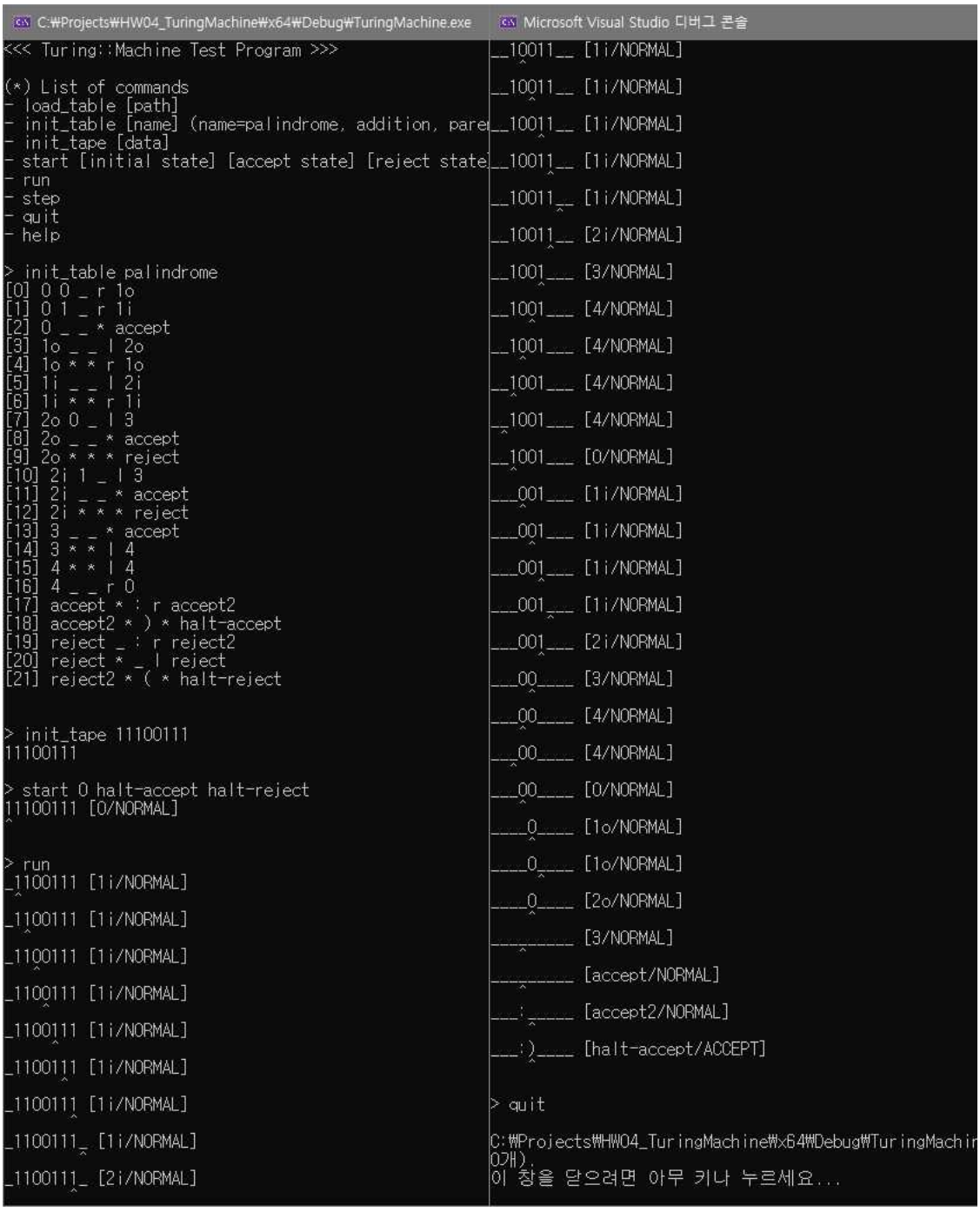
Table

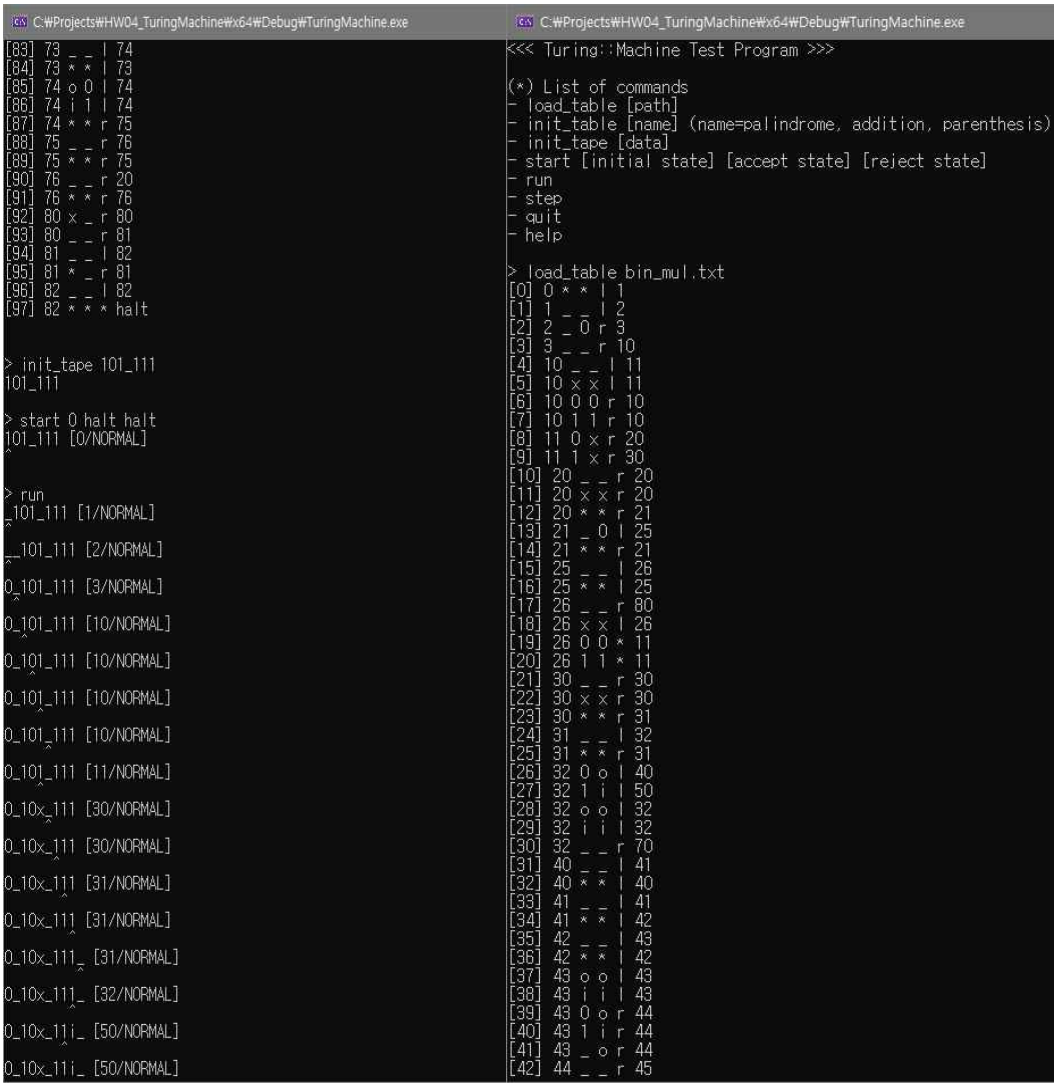

C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe
<<< Turing::Table Test Program >>>
(*) List of commands
- add [curr_s] [read_s] [write_s] [move] [next_s]
- find [curr_s] [read_s]
- initialize [name] (name = palindrome, addition, parenthesis
- load [path]
- clear
- print
- quit
- help
> add A 1 0 r B
[0] A 1 0 r B
> add xy K 9 H Sj
[0] A 1 0 r B
[1] xy K 9 * Sj
> clear
> print
> initialize addition
[0] 0 _ _ r 1
[1] 0 * * r 0
[2] 1 _ _ l 2
[3] 1 * * r 1
[4] 2 0 _ l 3x
[5] 2 1 _ l 3y
[6] 2 _ _ l 7
[7] 3x _ _ l 4x
[8] 3x * * l 3x
[9] 3y _ _ l 4y
[10] 3y * * l 3y
[11] 4x 0 x r 0
[12] 4x 1 y r 0
[13] 4x _ x r 0
[14] 4x * * l 4x
[15] 4y 0 1 * 5
[16] 4y 1 0 l 4y
[17] 4y _ 1 * 5
[18] 4y * * l 4y
[19] 5 x x l 6
[20] 5 y y l 6
[21] 5 _ _ l 6
[22] 5 * * r 5
[23] 6 0 x r 0
[24] 6 1 y r 0
[25] 7 x 0 l 7
[26] 7 y 1 l 7
[27] 7 _ _ r halt
[28] 7 * * l 7
> find 4x 1
4x 1 y r 0
> find 4x *
4x 0 x r 0
> load beaver_4.txt
[0] 0 * * * a
[1] a _ 1 r b
[2] a 1 1 l b
Microsoft Visual Studio 디버그 콘솔
[20] 5 y y l 6
[21] 5 _ _ l 6
[22] 5 * * r 5
[23] 6 0 x r 0
[24] 6 1 y r 0
[25] 7 x 0 l 7
[26] 7 y 1 l 7
[27] 7 _ _ r halt
[28] 7 * * l 7
> find 4x 1
4x 1 y r 0
> find 4x *
4x 0 x r 0
> load beaver_4.txt
[0] 0 * * * a
[1] a _ 1 r b
[2] a 1 1 l b
[3] b _ 1 l a
[4] b 1 _ l c
[5] c _ 1 r halt
[6] c 1 1 l d
[7] d _ 1 r d
[8] d 1 _ r a
> load bin_dec.txt
[0] 0 * * * 1
[1] 1 _ _ r 1
[2] 1 * * r 1a
[3] 1a * * r 1a
[4] 1a _ _ l 2
[5] 1b _ _ r 1
[6] 1b * * r 1b
[7] 2 1 0 l 3
[8] 2 0 1 l 2
[9] 2 _ _ r 20
[10] 3 * * l 3
[11] 3 _ _ l 4
[12] 4 0 1 r 1b
[13] 4 1 2 r 1b
[14] 4 2 3 r 1b
[15] 4 3 4 r 1b
[16] 4 4 5 r 1b
[17] 4 5 6 r 1b
[18] 4 6 7 r 1b
[19] 4 7 8 r 1b
[20] 4 8 9 r 1b
[21] 4 9 0 l 4
[22] 4 _ 1 r 1b
[23] 20 _ _ l 21
[24] 20 * _ r 20
[25] 21 _ _ l 21
[26] 21 * * l 21a
[27] 21a * * l 21a
[28] 21a _ _ r halt
> quit
C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe(프
이 창을 닫으려면 아무 키나 누르세요...

Tape


C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe
<<< Turing::Tape Test Program >>>
(*) List of commands
- construct
- destroy
- randomize
- initialize [s]
- read [i]
- write [i] [c]
- push_back [c]
- push_front [c]
- extend_right [n]
- exted_left [n]
- print
- quit
- help
> construct
> initialize 1000110
1000110
> read 2
0
> write 2 1
1010110
> write 3 x
101x110
> push_back 1
101x1101
> push_back y
101x1101y
> push_front 0
0101x1101y
> push_front z
z0101x1101y
> extend_right 10
z0101x1101y_
z0101x1101y__
z0101x1101y___
z0101x1101y____
z0101x1101y_____
z0101x1101y______
z0101x1101y_______
z0101x1101y________
z0101x1101y_________
z0101x1101y__________
> extend_left 10
_z0101x1101y__________
__z0101x1101y_________
___z0101x1101y________
____z0101x1101y_______
_____z0101x1101y______
______z0101x1101y_____
Microsoft Visual Studio 디버그 콘솔
1010110
> write 3 x
101x110
> push_back 1
101x1101
> push_back y
101x1101y
> push_front 0
0101x1101y
> push_front z
z0101x1101y
> extend_right 10
z0101x1101y_
z0101x1101y__
z0101x1101y___
z0101x1101y____
z0101x1101y_____
z0101x1101y______
z0101x1101y_______
z0101x1101y________
z0101x1101y_________
z0101x1101y__________
> extend_left 10
_z0101x1101y__________
__z0101x1101y_________
___z0101x1101y________
____z0101x1101y_______
_____z0101x1101y______
______z0101x1101y_____
_______z0101x1101y____
________z0101x1101y___
_________z0101x1101y__
__________z0101x1101y_
> randomize
jXMu^GuKsoVPXHl₩fuNk₩oQfC]ViyBv
> read 1
X
> write 1 k
jkMu^GuKsoVPXHl₩fuNk₩oQfC]ViyBv
> destroy
> quit
C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe(프
이 창을 닫으려면 아무 키나 누르세요...

Machine


C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe
<<< Turing::Machine Test Program >>>
(*) List of commands
- load_table [path]
- init_table [name] (name=palindrome, addition, pare
- init_tape [data]
- start [initial state] [accept state] [reject state
- run
- step
- quit
- help
> init_table palindrome
[0] 0 0 _ r 1o
[1] 0 1 _ r 1i
[2] 0 _ _ * accept
[3] 1o _ _ l 2o
[4] 1o * * r 1o
[5] 1i _ _ l 2i
[6] 1i * * r 1i
[7] 2o 0 _ l 3
[8] 2o _ _ * accept
[9] 2o * * * reject
[10] 2i 1 _ l 3
[11] 2i _ _ * accept
[12] 2i * * * reject
[13] 3 _ _ * accept
[14] 3 * * l 4
[15] 4 * * l 4
[16] 4 _ _ r 0
[17] accept * : r accept2
[18] accept2 *) * halt-accept
[19] reject _ : r reject2
[20] reject * _ l reject
[21] reject2 * (* halt-reject
> init_tape 11100111
11100111
> start 0 halt-accept halt-reject
11100111 [0/NORMAL]
> run
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111 [1i/NORMAL]
_1100111_ [1i/NORMAL]
_1100111_ [2i/NORMAL]
Microsoft Visual Studio 디버그 콘솔
__10011__ [1i/NORMAL]
__10011__ [1i/NORMAL]
__10011__ [1i/NORMAL]
__10011__ [1i/NORMAL]
__10011__ [1i/NORMAL]
__10011__ [2i/NORMAL]
__1001___ [3/NORMAL]
__1001___ [4/NORMAL]
__1001___ [4/NORMAL]
__1001___ [4/NORMAL]
__1001___ [4/NORMAL]
__1001___ [0/NORMAL]
___001___ [1i/NORMAL]
___001___ [1i/NORMAL]
___001___ [1i/NORMAL]
___001___ [1i/NORMAL]
___001___ [2i/NORMAL]
___00____ [3/NORMAL]
___00____ [4/NORMAL]
___00____ [4/NORMAL]
___00____ [0/NORMAL]
____0____ [1o/NORMAL]
____0____ [1o/NORMAL]
____0____ [2o/NORMAL]
_________ [3/NORMAL]
_________ [accept/NORMAL]
___:_____ [accept2/NORMAL]
___:)____ [halt-accept/ACCEPT]
> quit
C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachi
0개).
이 창을 닫으려면 아무 키나 누르세요...


C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe
[83] 73 _ _ l 74
[84] 73 * * l 73
[85] 74 o O l 74
[86] 74 i 1 l 74
[87] 74 * * r 75
[88] 75 _ _ r 76
[89] 75 * * r 75
[90] 76 _ _ r 20
[91] 76 * * r 76
[92] 80 x _ r 80
[93] 80 _ _ r 81
[94] 81 _ _ l 82
[95] 81 * _ r 81
[96] 82 _ _ l 82
[97] 82 * * * halt
> init_tape 101_111
101_111
> start 0 halt halt
101_111 [0/NORMAL]
> run
_101_111 [1/NORMAL]
__101_111 [2/NORMAL]
0_101_111 [3/NORMAL]
0_101_111 [10/NORMAL]
0_101_111 [10/NORMAL]
0_101_111 [10/NORMAL]
0_101_111 [10/NORMAL]
0_101_111 [11/NORMAL]
0_10x_111 [30/NORMAL]
0_10x_111 [30/NORMAL]
0_10x_111 [31/NORMAL]
0_10x_111 [31/NORMAL]
0_10x_111_ [31/NORMAL]
0_10x_111_ [32/NORMAL]
0_10x_11i_ [50/NORMAL]
0_10x_11i_ [50/NORMAL]
C:₩Projects₩HW04_TuringMachine₩x64₩Debug₩TuringMachine.exe
<<< Turing::Machine Test Program >>>
(*) List of commands
- load_table [path]
- init_table [name] (name=palindrome, addition, parenthesis)
- init_tape [data]
- start [initial state] [accept state] [reject state]
- run
- step
- quit
- help
> load_table bin_mul.txt
[0] 0 * * l 1
[1] 1 _ _ l 2
[2] 2 _ 0 r 3
[3] 3 _ _ r 10
[4] 10 _ _ l 11
[5] 10 x x l 11
[6] 10 0 0 r 10
[7] 10 1 1 r 10
[8] 11 0 x r 20
[9] 11 1 x r 30
[10] 20 _ _ r 20
[11] 20 x x r 20
[12] 20 * * r 21
[13] 21 _ 0 l 25
[14] 21 * * r 21
[15] 25 _ _ l 26
[16] 25 * * l 25
[17] 26 _ _ r 80
[18] 26 x x l 26
[19] 26 0 0 * 11
[20] 26 1 1 * 11
[21] 30 _ _ r 30
[22] 30 x x r 30
[23] 30 * * r 31
[24] 31 _ _ l 32
[25] 31 * * r 31
[26] 32 0 o l 40
[27] 32 1 i l 50
[28] 32 o o l 32
[29] 32 i i l 32
[30] 32 _ _ r 70
[31] 40 _ _ l 41
[32] 40 * * l 40
[33] 41 _ _ l 41
[34] 41 * * l 42
[35] 42 _ _ l 43
[36] 42 * * l 42
[37] 43 o o l 43
[38] 43 i i l 43
[39] 43 0 o r 44
[40] 43 1 i r 44
[41] 43 _ o r 44
[42] 44 _ _ r 45

요구 조건

Table

addTransition

class Table의 private 영역의 vector<Transition>타입 변수 transition에 입력받은 Transition 값을 push_back으로 추가하게 하였다.

print

for문으로 변수 transition의 size만큼 반복하여 출력하게 하였다.

clear

변수 transition을 vector의 clear함수로 비웠다.

findTransition

if문을 이용해 입력받은 값의 read가 *일 경우 for문으로 입력받은 state일 경우 찾아 반환하게 하여 처음 나오는 state를 반환하게 하였고, 그 외의 경우에는 state와 read가 같을 때 반환하게 하였다. 만약에 찾는 read가 없을 경우, read_s를 *로 설정하여 state와 read가 같은 transition을 찾아 반환하게 하였다.

initialize

우선, clear함수를 이용해 변수 transition에 있는 값들을 비웠다. Util::split을 이용해 문자가 '\n'일 때 문자열을 나눠

vector<string> 타입의 temp에 저장하도록 하였다.
for 문을 이용해 temp의 각 요소들에 접근하여 Util::stripComment 함수로 ;뒤의 문자열을 삭제하고, 접근한 요소가 비어있다면 그 요소를 삭제하도록 하였다.
그 다음 for문을 이용해 temp의 문자열을 분해하여 transition 변수에 추가하도록 하였다. Util::split 함수를 이용해 temp의 한 요소를 띄어쓰기마다 나누어주어 vector<string>타입의 변수 result에 저장해주었다. 그 다음 result의 각 요소에 접근하여 read값과 write값에 해당하는 요소를 string에서 char타입으로 바꿔주고 Move값에 해당하는 요소들을 Move타입으로 바꿔준 후, 변수 transition에 push_back으로 추가하게 하였다.

load

텍스트 파일을 읽어와 string 타입으로 한 변수에 저장 후 initialize 함수를 호출하게 하였다.

Tape

initialize

HEAP CORRUPTION DETECTED 에러 해결을 위해 먼저 reserve 함수를 호출해 입력한 string의 size만큼 공간을 배정해준다.
그리고 변수 sz를 입력한 string의 size로, 변수 space를 입력한 string size + 1로(HEAP CORRUPTION DETECTED 에러 방지) 설정한다. 그 후 strcpy로 변수 elem에 입력한 string값을 char값으로 변환하여 저장하게 하였다.

print

for문을 이용해 변수 elem의 각 요소들을 출력하게 하였다.

Tape::Tape(const Tape& t)

복사할 때 쓰는 constructor이므로 입력받은 값을 for문을 이용해 복사하도록 하였다.

Tape::Tape(Tape&& t)

이동할 때 쓰는 constructor이므로 입력받은 값을 초기화하도록 하였다.

Tape::operator=(const Tape& t)

만약 비교하는 값이 같을 경우 바로 자신의 값을 return하도록 하였다.
만약 '=' 우측 변수의 size가 '='좌측 변수의 space보다 같거나 작은 경우 변수 elem을 복사한 후 좌측변수의 size를 우측변수의 size로 바꿔준 후 return하게 하였다.

그 외의 경우에는 새로운 공간을 동적할당하여 복사한 후 return하게 하였다.

Tape::operator=(Tape&& t)

'=' 좌측 변수의 동적할당된 공간을 해체하고 우측 변수의 값들을 복사 한 후 우측변수의 동적할당된 공간을 해제하도록 하였다.

read

입력받은 변수 c에 변수 elem[i]의 값을 저장하도록 하였다.

write

변수 elem[i]에 입력받은 변수 c를 저장하도록 하였다.

read_for_Machine

Machine을 위해 만든 함수로, 입력받은 변수 c에 변수 elem[i]의 값을 저장하도록 하였다.

write_for_Machine

Machine을 위해 만든 함수로, 만약 입력받은 c가 *일 경우 elem[i]의 값을 바꾸지 않고, 그 외의 경우에 변수 elem[i]에 입력받은 변수 c를 저장하도록 하였다.

clear

변수 sz의 값을 0으로 바꾸게 하였다.

push_back

변수 sz의 값이 10일 경우 reserve 함수를 호출해 10의 공간을

할당하게 하였고, sz == space - 1(HEAP CORRUPTION DETECTED 방지)일 때 reverse 함수를 호출해 space의 5배의 공간을 할당하도록 하였다.
그 다음 elem의 마지막 인덱스에 입력된 c값을 넣어주고 sz값을 1 증가하도록 하였다.

push_front

변수 sz의 값이 10일 경우 reserve 함수를 호출해 10의 공간을 할당하게 하였고, sz == space - 1(HEAP CORRUPTION DETECTED 방지)일 때 reverse 함수를 호출해 space의 5배의 공간을 할당하도록 하였다.
그 다음 for 문을 이용해 elem의 요소들을 한 칸씩 뒤로 미룬 다음 elem[0]에 입력받은 c값을 넣어주고 sz값을 1 증가하도록 하였다.

reserve

HEAP CORRUPTION DETECTED 에러 방지를 위해 space가 0일 경우 입력된 newalloc+1값의 공간을 할당하도록 하였다.
newalloc<=space일 경우 바로 return되도록 하였다.
그 외의 경우에는 newalloc만큼의 크기를 동적할당하여 elem의 공간을 넓히도록 하였다.

resize

reserve 함수를 호출해 입력받은 값으로 공간을 할당하고, elem의 요소들을 입력받은 값의 갯수만큼 0으로 초기화 한 후 sz를 입력받은 값으로 바꾸게 하였다.

Machine

initTape

Tape의 initialize 함수를 불러와 tape 변수를 initialize 하도록 하였다.

initTable

Table의 initialize 함수를 불러와 table 변수를 initialize 하도록 하였다.

loadTable

Table의 load 함수를 불러와 table 변수에 load하도록 하였다.

start

current_state 변수에 입력된 start_state 변수값을 대입하고, accept_state 변수엔 입력된 accept_state 변수, reject_state 변수엔 reject_state 변수를 대입했다.
current_state==start_state라면 current_mode=:NORMAL,
current_state==accept_state라면 current_mode=ACCEPT,
current_state==reject_state라면 current_mode=REJECT,
그 외의 경우엔 current_mode=ERROR로 설정해줬다.

step

char 타입 변수 c에 현재 tape 위치에 해당하는 글자를 read_for_Machine을 이용해 저장한다. 만약, 현재위치가 tape의 size보다 크다면 읽을 수 있는 글자가 없으므로 c값에 '_'를 저장한다. Transition* 타입 변수 trans에 현재상태와 변수 c에 저장되어

있는 글자를 포함하고 있는 transition을 findTransition으로 찾아 trans에 저장한다.

현재 tape의 글자를 transtion에 따라 변경하는 것은 3가지로 나뉜다.

trans의 Move가 LEFT 이고 현재 위치가 0보다 작거나 같을 때

: trans의 write가 '*'이라면 현재 글자를 변경하지 않는 것이므로 push_front로 tape의 맨 앞에 '_'글자를 넣어 자리를 만들어 준다.

trans의 write가 '*'이 아니라면 현재 글자를 write_for_Machine으로 바꿔주고 tape의 맨 앞에 push_front로 '_'글자를 넣어 자리를 만들어 준다.

trans의 Move가 Right 이고 현재 위치가 tape의 size보다 1 작을 때(tape의 끝에 있을 때)

: trans의 write가 '*'이라면 현재 글자를 변경하지 않는 것이므로 push_back으로 tape의 맨 뒤에 '_'글자를 넣어 자리를 만들어 준다.

trans의 write가 '*'dI 아니라면 현재 글자를 write_for_Machine으로 바꿔주고 tape의 맨 뒤에 push_back으로 '_'글잘르 넣어 자리를 만들어 준다.

그 외

현재 글자를 write_for_Machine을 사용해 trans의 write로 바꿔준다.

현재 tape의 글자를 변경하는 것을 끝낸 후에, 현재 상태가 저장되어 있는 current_state의 값을 다음 상태인 trans의 nextstate로 변경한다.

만약 current_state가 accept_state 일 경우 current_mode를

ACCEPT로 바꿔준다. current_state가 reject_state일 경우에는 current_mode를 REJECT로 바꿔준다.
current_state가 accept_state거나 reject_state일 경우 Machine이 끝까지 다 작동했다는 뜻 이므로 return으로 함수를 종료한다.
만약 trans의 Move가 LEFT라면 현재 위치를 저장하고 있는 curren_pos의 값을 1 낮춘다.
만약 trans의 Move가 Right라면 현재 위치를 저장하고 있는 curren_pos의 값을 1 높인다.

Comment

코딩하면서 가장 어려웠던 부분은 동적 메모리 할당과 해제였다. 코드를 작성하고 수정할 때마다 튀어나오는 HEAP CORRUPTION DETECTED 문구는 나를 괴롭게 만들었다. 눈에 보이지 않는 메모리 영역에서 일어나는 일이다 보니 문제 원인을 찾는 것에 시간이 매우 많이 걸렸다. 코딩 시간의 70%가 HEAP CORRUPTION DETECTED와의 싸움이었다고 봐도 무방할 정도였다.
내게 일어난 HEAP CORRUPTION DETECTED 에러의 문제는 대부분 메모리를 할당하고 해제할 때 할당한 값보다 더 많이 데이터가 들어 있어서 생겼던 것으로 보인다. 예를 들어 할당된 메모리 공간이 5인 곳에 크기가 6짜리인 무언가가 들어가 있어서 이를 해제 할 때 HEAP CORRUPTION DETECTED 에러가 생겼다. 사실 이를 해결하는 방법은 생각보다 간단했다. 메모리 공간을 동적 할당할 때 무작정 큰 값을 할당하면 되는 것이었다. 물론 필요한 값만큼 할당하는 것보다는 메모리 효율 측면에서 비효율적이다. 하지만 필요한 값만큼 할당하는 것은 생각보다 어렵고 복잡했다. 필요한 크기만큼 할당하기 위해 오랜 시간을 투자했지만 완벽하게 해결해내지 못했다. 그래서 생각해 낸 대안이 충분히 큰 값을 할당하는 것이었다.
첫 번째 문제는 default constructor에서 발생했다. Tape의 default constructor에서 elem의 size, space 값을 받지 않고 0으로 초기화 하다 보니 에러가 생겼다. 그래서 reserve 함수에 space 값이 0일 경우 실행해야 할 코드를 따로 만들어 해결했다.
두 번째 문제는, extend_left와 extend_right 할 때도 에러가 났었다. 이는 push_back과 push_front에서 reserve를 사용해 더

넓은 공간을 할당할 때 충분히 넓은 공간을 할당하지 않아서 생기는 문제로 확인되었다. 그래서 push_back과 push_right에서 reserve 함수를 호출할 때 충분히 큰 값을 줌으로써 해결할 수 있었다.
세 번째 문제는 Machine을 가동할 때 생겼다. step을 사용할 때 현재 위치가 tape 문자열의 맨 앞이지만 다음 transition이 그 앞에서 작업할 때, 혹은 그 반대로 tape 문자열의 맨 뒤지만 다음 transition이 그 뒤에서 작업할 때 발생했다. 이는 위와 같은 상황이 발생했을 때 앞이나 뒤에 문자 '_'를 넣어 공간을 만들어 줌으로써 해결할 수 있었다.

이번 과제를 하면서 이미 개발된 코드들을 활용할 수 있으면 활용하는 게 좋다고 하신 교수님 말씀이 생각났다. 보이지 않는 메모리를 직접 관리하는 것은 생각보다 어려웠다. vector가 얼마나 좋은 기능인지 다시 보게 되었다.