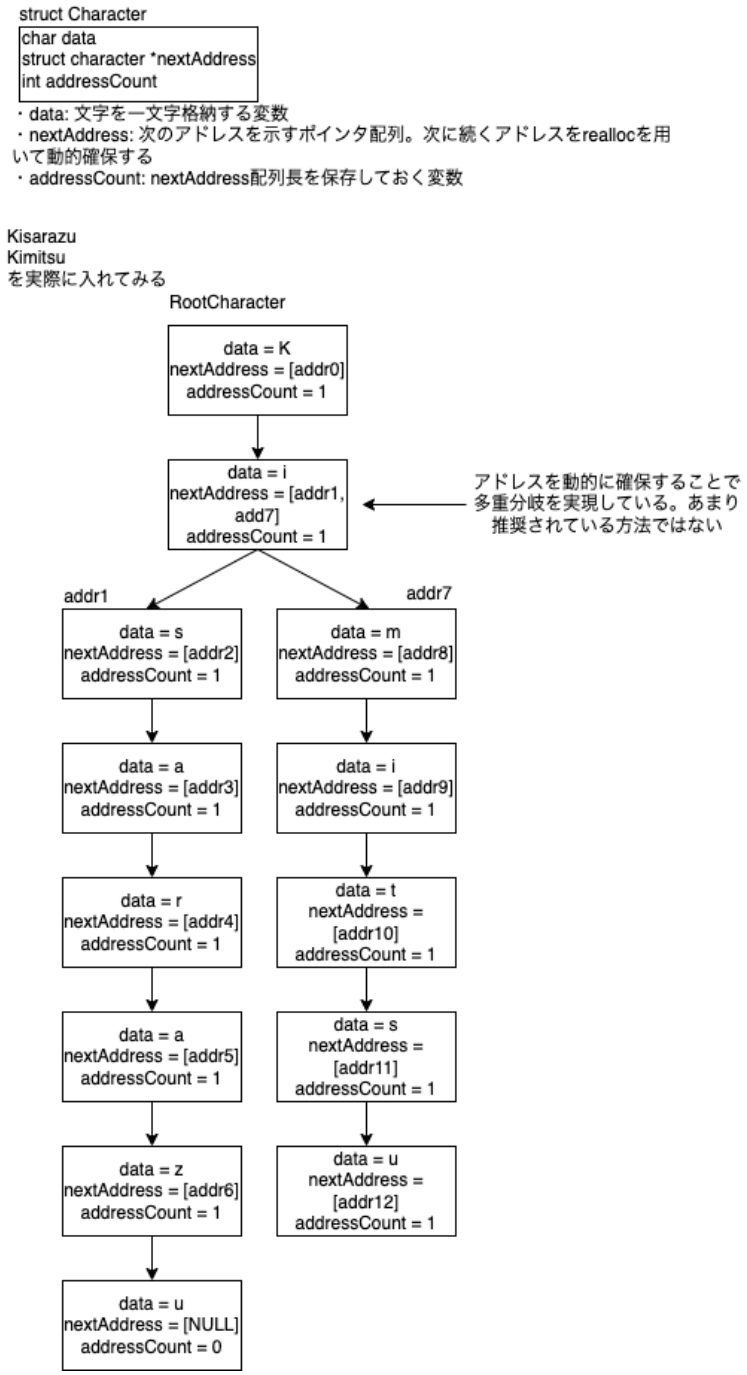
プログラミング演習 IIB（グループワーク）報告書

20-413 Group-2 北野正樹

① 自分でプログラムを作成。

まず最初に自分でプログラムを作成した。作成したプログラムのアルゴリズムを以下に示す。



このアルゴリズムは nextAddress というポインタを動的確保することで多重分岐に対応している。しかし、realloc による再動的確保はあまり好ましい選択ではなく、大きな分岐があった際の可用性は不明だ。

作成したプログラムを以下に示す。

```
#include <stdio.h>
#include <stdlib.h>

typedef struct character {
   char data;
   struct character *nextAddress;
   int addressCount;
} Character;

Character *rootCharacter = NULL;

Character* searchEndCharacter(Character *c, char ch);
void add(Character *c);

int main(void) {
   char keyBoardInput[256];
   while (1) {
      printf("Type Your Key: ");
      scanf("%s", keyBoardInput);
      if (keyBoardInput[0] == 'q' && keyBoardInput[1] == '¥0') {
         return 0;
      }
      for (int i = 0;keyBoardInput[i] != '¥0'; i++) {
         Character *newCharacter;
         newCharacter = (Character*) malloc(sizeof(Character));
         newCharacter -> data = keyBoardInput[i];
         newCharacter -> nextAddress = NULL;
                        newCharacter -> addressCount = 0;
         if (rootCharacter == NULL) {
                           rootCharacter = newCharacter;
                           printf("put the [%c] as root¥n", keyBoardInput[i]);
```

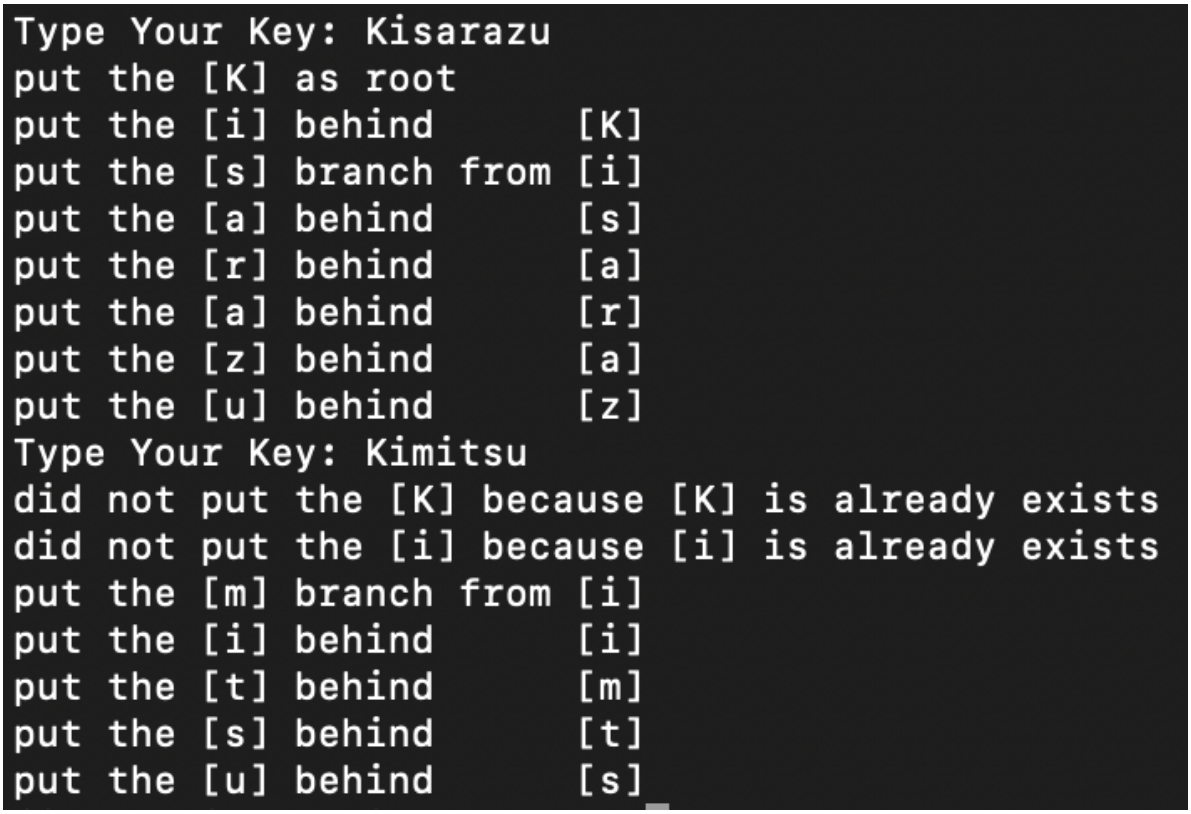
```
                        continue;
        } else {
                        Character *searchResults = searchEndCharacter(rootCharacter,
keyBoardInput[i]);
                        if (searchResults -> nextAddress == NULL) {
                                searchResults -> nextAddress = newCharacter;
                                searchResults -> addressCount++;
                                printf("put the [%c] behind [%c]¥n", keyBoardInput[i],
searchResults -> data);
                        } else {
                                Character* tmp = (Character*) realloc(searchResults ->
nextAddress, sizeof(Character));
                                if (tmp == NULL) {
                                        printf("memory error¥n");
                                        free(searchResults);
                                        return -1;
                                } else {
                                        printf("put the [%c] branch from [%c]¥n",
keyBoardInput[i], searchResults -> data);
                                        searchResults -> nextAddress = tmp;
                                        searchResults -> addressCount++;
                                }
                        }
        }
      }
   }
   free(rootCharacter);
   return 0;
}

Character* searchEndCharacter(Character *c, char ch) {
   if (c -> nextAddress == NULL || c -> addressCount == 0) {
            return c;
   } else {
            for (int i = 0; i < c -> addressCount; i++) {
                    if (c -> nextAddress[i].data == ch) {
```

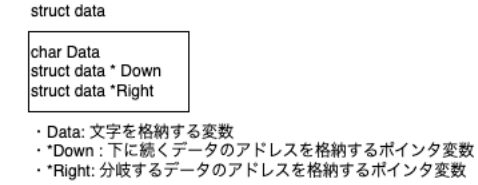
```
                searchEndCharacter(&c -> nextAddress[i], ch);
            }
        }
        return c;
    }
}
```

実行結果を以下に示す。

② みんなのコードやアルゴリズムを比較して最適なアルゴリズムとプログラムを作成する。

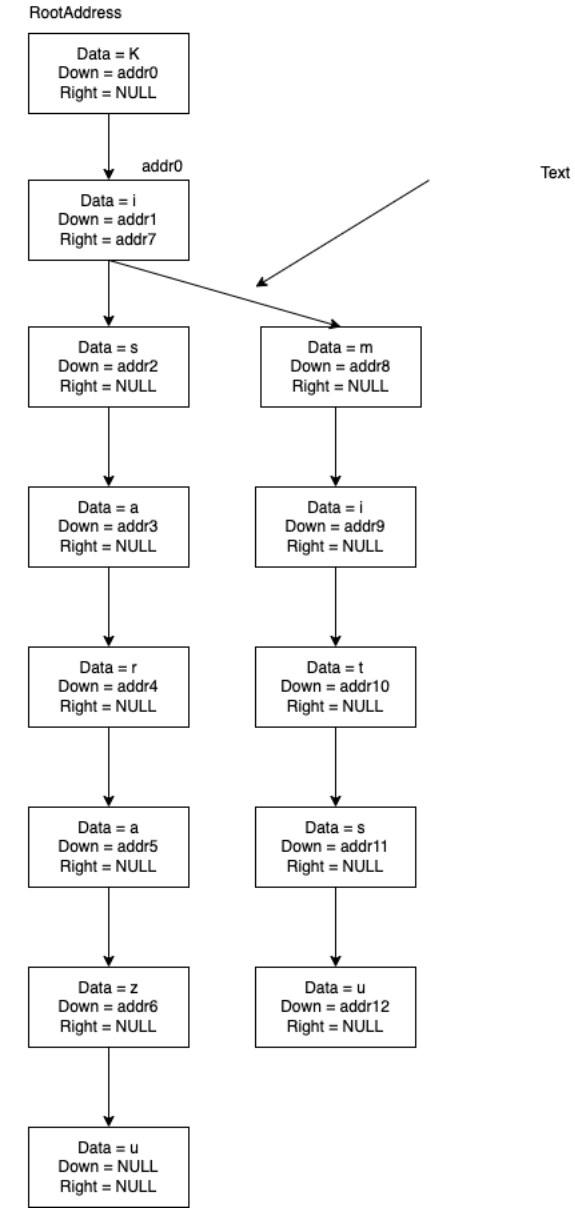
みんなで話し合った結果、自分のプログラムだとやはり安定性に欠けるということで別のアプローチを考えた。班で話し合った結果のアルゴリズムを以下に示す。





・Data: 文字を格納する変数
・*Down : 下に続くデータのアドレスを格納するポインタ変数
・*Right: 分岐するデータのアドレスを格納するポインタ変数

Kisarazu
Kimitsu
を実際に入れてみる



このアルゴリズムにしたことで、メモリの再動的割り当てがなくなり、安定したメモリ管理ができるようになった。また、割り当ての時間がなくなったことで処理が高速化した。

このアルゴリズムを適用したプログラムを以下に示す。

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>

#define MAX 100

typedef struct data
{
   char Data;
   struct data *Down;
   struct data *Right;
}Data;

void AddWord();
void SearchAndInsert(Data* Node);
void DisplayNumbers(Data* Node);
void Travel(Data* Node,int n);
Data *NewData, *CurrentData, *Root=NULL;
char NewWord[100];
int pos;

int main()
{

 char KeyBoardInput,Options;
while(1)
   {
   printf("A: Add Word, D:Display, Q: Quit¥n");
   printf("Type desired options(A,D,Q):");
   scanf("%s",&KeyBoardInput);
   Options=toupper(KeyBoardInput);
   switch (Options)
    {
      case 'A':
      AddWord();
      break;
```

```
        case 'D':
        printf("Inserted Numbers are as follows¥n");
        DisplayNumbers(Root);
        printf("¥n");
        break;
        case 'Q':
        return 0;
        default:
        printf("Invalid Option.¥n");
        break;
        }
    }
}

void AddWord()
{
    printf("単語を入力してください:¥n");
    scanf("%s",&NewWord);
    if (Root==NULL)
     {
        int i = 0;
        NewData=(Data*)malloc(sizeof(Data));
        NewData->Data=NewWord[i];
        NewData->Right=NULL;
        NewData->Down=NULL;
        Root=NewData;
        CurrentData = NewData;

        for(i=1;NewWord[i]!='¥0'&&pos<MAX;i++){
           NewData=(Data*)malloc(sizeof(Data));
           NewData->Data=NewWord[i];
           NewData->Right=NULL;
           NewData->Down=NULL;
           CurrentData->Down=NewData;
           CurrentData=NewData;
        }
```

```
    }
      else
      {
        pos=0;
        SearchAndInsert(Root);
      }
}

void SearchAndInsert(Data *Node)
{
    if((NewWord[pos] != Node->Data)&& (Node->Right!=NULL))
      {
        SearchAndInsert(Node->Right);
      }
    else if((NewWord[pos] != Node->Data) && (Node->Right==NULL))
      {
        NewData=(Data*)malloc(sizeof(Data));
        Node->Right=NewData;
        NewData->Data=NewWord[pos];
        NewData->Right=NULL;
        CurrentData = NewData;

        for(pos++;NewWord[pos]!='¥0'&&pos<MAX;pos++){
          NewData=(Data*)malloc(sizeof(Data));
          NewData->Data=NewWord[pos];
          NewData->Right=NULL;
          NewData->Down=NULL;
          CurrentData->Down=NewData;
          CurrentData=NewData;
        }
      }
    else if((NewWord[pos]!=Node->Data)&&(Node->Down==NULL)){
       NewData=(Data*)malloc(sizeof(Data));
        Node->Down=NewData;
        NewData->Data=NewWord[pos];
        NewData->Right=NULL;
```

```
            CurrentData = NewData;

            for(;NewWord[pos]!='¥0'&&pos<MAX;pos++){
               NewData=(Data*)malloc(sizeof(Data));
               NewData->Data=NewWord[pos];
               NewData->Right=NULL;
               NewData->Down=NULL;
               CurrentData->Down=NewData;
               CurrentData=NewData;
            }
      }
        else{
          if((NewWord[pos]!='¥0')&&(Node->Down!=NULL)){
          pos++;
          SearchAndInsert(Node->Down);
          }
          else{
             printf("all match");
             return;
          }
        }
}

void   DisplayNumbers(Data *Node)
{
   if (Root==NULL)
     {
      printf("No data exists¥n");
      return;
     }
     else
     {
        Travel(Root,0);
     }
}
```

```
        void Travel(Data *Node,int n){
          printf("%c",Node->Data);
          if(Node->Down!=NULL){
            Travel(Node->Down,n+1);
          }
          else{
            printf("¥n");
          }

          if(Node->Right!=NULL){
            for(int i=0;i<n-1;i++){
            printf(" ");
            }
            printf("~");
            Travel(Node->Right,n);
          }
}
```

実行結果を以下に示す。