プログラミング演習 第 3 回

問 A [03A](出席メール): メモリ配置の調査

図 1 のプログラムにおける大域変数 gi，gc，gd，ga，ga[0]，ga[1]，ga[2]，...，gs，gs.i，gs.c，gs.d，局所変数 mi，mc，md，ma，ma[0]，ma[1]，ma[2]，...，ms，ms.i，ms.c，ms.d のアドレス，および malloc で用意された領域のアドレス (p と q が保持)，さらに main から直接呼び出されたときの f1 の変数 i，f2 経由で呼び出されたときの f1 の変数 i，f3 経由で呼び出されたときの f1 の変数 i のアドレスを調べ比較し，各種の変数がメモリ中でどのように並んでいるか調査し考察せよ．必要に応じて変数を増やしたり，構造体や配列の型や内容を変更してみるとよい．変数のアドレスは変数にアドレス演算子 & を前置すれば得られ，アドレスの値を 16 進数として表示するには printf の書式に%x あるいは %p を指定すればよい．なお，本課題は実質的なプログラミングを伴わないため，調査して分かったことや考察を，ただのテキストとしてファイルに保存し，それをメールで提出すればよい．

```
struct mydata {int i; char c; double d;};
int gi; char gc; double gd;
int ga[100];
struct mydata gs;
void f1(void) {
  int i;
  /* ここで，局所変数 i のアドレスを表示．*/
}
void f2(void) {
  int a[10];  /* この下で，この a のアドレスも表示するとよい．*/
  f1();
}
void f3(void) {
  int a[20];  /* この下で，この a のアドレスも表示するとよい．*/
  f1();
}
int main(void) {
  int mi; char mc; double md;
  int ma[100];
  struct mydata ms;
  char *p, *q;
  p = (char*)malloc(100);
  q = (char*)malloc(100);
  /* ここで，大域変数，main の局所変数，malloc() で確保したアドレスを表示．*/
  f1();
  f2();
  f3();
  return 0;
}
```

図 1: メモリ配置の調査対象となるプログラム

問 B [03B]: 部分文字列の探索

引数として文字列へのポインタが二つ与えられたとき，第一引数で示される文字列の中から，第二引数で示される文字列を前から探し，あればそのポインタを，なければ NULL ポインタを返す関数 findhead を作成せよ．さらにこの関数を使い，端末から二つの文字列を読み込み，最初の文字列から二番目の文字列を探し，その位置以降を出力するプログラムを作成せよ．端末から与えられる文字列は 78 文字以下で ASCII 文字のみで構成されると仮定してよい．findhead のプロトタイプ宣言は次のとおりである．

```
char *findhead(char *s1, char *s2);
```

また以下の二点をプログラミング上の条件とする．

- あらかじめ用意されている同等の文字列操作関数 (たとえば strstr) などを呼び出してはいけない．
- 与えられたポインタ (s1，s2) をそのまま用い，配列の添字式 (たとえば s1[i]) は用いてはいけない．

入出力例

入力 1

```
exciting university
it
```

入力 1 に対応する出力

```
iting university
```

入力 2

```
exciting university
bore
```

入力 2 に対する出力

(空行)

問 C [03C]: ポインタを介した構造体の操作

p.4 に示す図 2 は，端末から 5 人分の学籍番号 (数値) とアカウント名 (8 文字の文字列) の組を受けとって記憶し，その後に端末から入力される学籍番号が記憶されていればそれに対応したアカウント名を，記録されていなければ no data を表示するプログラムである．ここでは学籍番号とアカウント名の組の記憶には，構造体 struct student を用いている．main から呼び出される関数 set と search を定義してプログラムを完成させよ．ただし set は，第二引数の学籍番号と第三引数のアカウント名を，第一引数で示される struct student のメンバーとしてセットする．また search は，第一引数の struct student の配列 (配列の大きさは第二引数で与えられる) から，第三引数の学籍番号を持つ要素を探し，あればその要素へのポインタを，なければ NULL ポインタを返す．

入出力例

入力

```
2013001 aa013001
2013002 bb013002
2013003 mm013003
2013004 nn013004
2013005 zz013005
2013004
2014003
2013001
```

出力

```
nn013004
no data
aa013001
```

問 D [03D]: 問 C の改良

プログラムの実行時に struct student を蓄える領域を必要に応じて確保していくことで，メモリがある限り学籍番号とアカウント名を記憶できるように，問 C のプログラムを改造せよ．

ヒント

- malloc を用いてプログラム実行中に struct student の配列を用意し，学籍番号とアカウント名をセットしていく．配列の大きさ以上の入力が与えられた場合は，さらに大きな (たとえば二倍の大きさの) 配列を新たに用意しそれまでの入力をコピーすればよい．コピー後は元の配列の領域を free で解放する．(領域の再確保とコピー/解放をまとめて行なう realloc を用いてもよい) このとき領域が確保されない場合 (malloc, realloc の返り値が NULL) はその場で停止するようにせよ．

  なお，(main の前半の) 学籍番号とアカウント名の組を登録する部分は，固定回数の for ループで表現せずに，空行等で登録を終了するようにする．あるいは、入力行の内容に応じて、(学籍番号とアカウント名ならば) 登録 (set)、(学籍番号のみならば) 検索 (search) を行うようにしてもよい。

- 学籍番号とアカウント名をセットする関数 (set) と，学籍番号に対応する struct student を探す関数 (search) はそのまま利用できる．また学籍番号が与えられる間，対応するアカウント名を表示する部分 (main の後半) もほぼ変更する必要はない．

```
#include <stdio.h>
#include <string.h>
struct student {
  int  id;          /* 学籍番号 */
  char account[9];  /* アカウント名 */
};
void set(struct student *s, int id, char *account) {
  /* ここを定義する. */
}
struct student *search(struct student students[], int n, int id) {
  /* ここを定義する. */
}

int main(void) {
#define BUFSIZE 80
#define STDNT   5
  struct student students[STDNT], *s;
  char buf[BUFSIZE], *account;
  int i, len, id;
  /* 5(STDNT) 人分の学籍番号とアカウント名を読み込み, 配列 students にセットする. */
  for (i = 0; i < STDNT; i++) {
    fgets(buf, sizeof(buf), stdin);
    len = strlen(buf);
    if (buf[len - 1] == '\n') buf[len - 1] = '\0';
    buf[7] = '\0';
    id = atoi(buf);
    account = buf + 8;
    while (*account == ' ') {account++;}
    set(&students[i], id, account);   /* ここから set を呼び出している */
  }
  /* 学籍番号が与えられる間, それに対応するアカウント名を表示する. */
  while (fgets(buf, sizeof(buf), stdin)) {
    if (buf[0] == '\n' || buf[0] == '\0') break;
    s = search(students, STDNT, atoi(buf));  /* ここから search を呼び出している */
    if (s == NULL)
      printf("no data\n");
    else
      printf("%s\n", s->account);
  }
  return 0;
}
```

図 2: 5 人分の学籍番号とアカウント名を記憶後, 学籍番号に対応するアカウント名を表示するプログラム