Q2 課題 「ポインタ」

湯素華 (CS プログラム)*

2018 年 4月 16 (I3)・19 (I1)・20 (I2) 日

1　まず出席確認

/ced-home/staff/q/q2/shusseki-iN を実行せよ。ここで、N はクラス番号 $N \in \{1, 2, 3\}$ である。一応やり方は、端末エミュレータで

```
[ya000836@purple99 ~/]$ /ced-home/staff/q/q2/shusseki-iN
```

である。

2　次に準備

自分のホームディレクトリ[1]の下に q2 というディレクトリを作成し、その下に移動せよ。やり方は、端末エミュレータで

```
[ya000836@purple99 ~/]$ mkdir q2
[ya000836@purple99 ~/]$ cd q2
[ya000836@purple99 ~/q2]$
```

という感じ。

次に、/ced-home/staff/q/q2 の下にある q2-で始まるファイル名のファイルを全てコピーせよ。以下のようにすれば良い。

```
[ya000836@purple99 ~/q2]$ cp /ced-home/staff/q/q2/q2-* .
```

3　Q2 課題

この授業では、異なる型の変数に対応するポイントの操作を練習する。

q2-1：ポインタの演算（初級課題）

基本データ（int、float など）と同じように、ポインタは型があり、異なる型のポインタ間の変換も可能である。ただし、下記のことに気をつけてください。

1. 「ポインタに 1 加える」ということは、「ポインタに格納されているアドレス＋ 1 番地」ではなく、「ポインタに格納されているアドレス＋型のサイズ」である。
2. 「ポインタに格納されているアドレスが同じでも、型が変わると、そのアドレスを使って取得する値が変わる」。

*shtang@uec.ac.jp. 研究室ホームページは http://www.tang.cs.uec.ac.jp/

[1] まあどこでもいいけど、特にこだわりがなければ$HOME で。

1. 図 1 に示すように、cptr を char*型のポイントとして使って配列 x を初期化してから、そのポイントを float*型のポインタとして解釈する際、入力の数字 num に対応する配列要素のアドレスが

```
ptrnew = ((float*)ptr) + num
```

で算出され、その内容が

```
value  = *((float*)cptr+num)
```

で取得できます。

```
int main(int argc, char *argv[]) {
    long   x[100];
    char  *typeStr = argv[1];
    void  *ptr = (void*)strtol(argv[2], NULL, 16);
    int    num = atoi(argv[3]);
    char  *cptr= (char*)x;
    int i;
    for (i=0; i<100*sizeof(long); i++)
        cptr[i] = i + 0x10;
    // example
    if (!strcmp(typeStr, "float")) {
        float *ptrnew;
        float value;
        ptrnew = ((float*)ptr) + num;
        value  = *((float*)cptr+num);
        printf("%p 0x%x\n", ptrnew, value);
    }
    return 0;
}
```

図 1: ポインタの操作例

2. 図 2 に示すコードのひな形に従い、これを埋めて、char*型、short*型、int*型、long*型などのポイントの操作コードを完成せよ。

3. このプログラムを完成してコンパイルすると、以下のように実行することが可能になる。

```
[ya000836@purple99 ~/q2]$ gcc -Wall -std=c99 -o q2-1 q2-1.c
[ya000836@purple99 ~/q2]$ ./q2-1 char 0x10000 2
0x10002 0x12
[ya000836@purple99 ~/q2]$ ./q2-1 short 0x10000 2
0x10004 0x1514
[ya000836@purple99 ~/q2]$ ./q2-1 int 0x10000 2
0x10008 0x1b1a1918
[ya000836@purple99 ~/q2]$ ./q2-1 long 0x10000 2
0x10010 0x2726252423222120
[ya000836@purple99 ~/q2]$
```

4. もし関数が正しく書けたと確信したら、以下のようにしてプログラムの動作をチェックせよ。

```
if (!strcmp(typeStr, "char")) {
    char *ptrnew;
    char value;
    // add your code here
    printf("%p 0x%x\n", ptrnew, value);
}
else if (!strcmp(typeStr, "short")) {
    short *ptrnew;
    short value;
    // add your code here
    printf("%p 0x%x\n", ptrnew, value);
}
else if (!strcmp(typeStr, "int")) {
    int  *ptrnew;
    int  value;
    // add your code here
    printf("%p 0x%x\n", ptrnew, value);
}
else if (!strcmp(typeStr, "long")) {
    long *ptrnew;
    long value;
    // add your code here
    printf("%p 0x%lx\n", ptrnew, value);
}
```

図 2: ソースコード q2-1.c

```
[ya000836@purple99 ~/q2]$ /ced-home/staff/q/q2/check1-iN q2-1.c
```

ここで、N は自分のクラス番号 ($N \in \{1, 2, 3\}$) である。check1-iN は、適当なテストケースをランダムに発生させて動作をチェックするプログラムである。動作が正しければ、「正解」と出力して加点する。間違っていた場合は、どういうテストケースでどう間違えたかを出力する。

チェックプログラムは何回実行してもかまわない。好きなだけ試すこと。

5. なお、自分の現在の点数は

```
[ya000836@purple99 ~/q2]$ /ced-home/staff/q/q2/score-iN
```

で調べられる。同様に N は自分のクラス番号 ($N \in \{1, 2, 3\}$) である。

q2-2：部分文字列の探索（初級課題）

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 findfirst を使い、端末から読み込んだ二つの文字列について、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。ただし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。

図 3 に示すコードのひな形に従い、これを埋めて、findfirst 関数 (str と sub がさしている文字が同じ場合、sub の残りの文字は str に含まれているかのチェック) を完成せよ。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群 strxxx など (たとえば strstr や strcmp) 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。

```
char *findfirst(char*str, char*sub) {
    char *a, *b;
    if (*sub == 0) {
        return str;
    }
    for (; *str != 0; str += 1) {
        if (*str != *sub) {
            continue;
        }
        // add your code here
        // do not use any functions
    }
    return NULL;
}
```

図 3: ソースコード q2-2.c

このプログラムを完成してコンパイルすると、以下のように実行することが可能になる。

```
[ya000836@purple99 ~/q2]$ gcc -Wall -std=c99 -o q2-2 q2-2.c
[ya000836@purple99 ~/q2]$ ./q2-2 exiciting it
iting
[ya000836@purple99 ~/q2]$ ./q2-2 exiciting bore

[ya000836@purple99 ~/q2]$
```

図 4: 実行例 q2-2

q2-3：ポインタの型によるデータの異なる解釈（中級課題）

課題 1 ではポインタの型変換操作を行ったが、ここでは、下記のように、ポインタの型変換を利用して浮動小数点数 float_num(float や double) を整数 integer として解釈し、その $\log_{10}$ 値を近似せよ。

```
float float_num;
int *iPtr, integer;
iPtr = (int*) &float_num;
integer = *iPtr;
```

単精度の浮動小数点数（float）は、32 ビットの内、$b_{22}b_{21}\cdots b_0$ は小数部分であり、$b_{30}b_{29}\cdots b_{23}$ は指数部分であり、b_{31} は符号である。その値は、$(-1)^{b_{31}}\times(1.b_{22}b_{21}\cdots b_0)_2\times 2^{(b_{30}b_{29}\cdots b_{23})_2-127}$ である。ただし、$(b_{n+m-1}b_{n+m-2}\cdots b_n)_2$ は二進 m 桁の整数である。

倍精度の浮動小数点数（double）は、64 ビットの内、$b_{51}b_{50}\cdots b_0$ は小数部分であり、$b_{62}b_{61}\cdots b_{52}$ は指数部分であり、b_{63} は符号である。その値は、$(-1)^{b_{63}}\times(1.b_{51}b_{50}\cdots b_0)_2\times 2^{(b_{62}b_{61}\cdots b_{52})_2-1023}$ である。

浮動小数点数の指数部分（float の場合 $(b_{30}b_{29}\cdots b_{23})_2 - 127$、double の場合 $(b_{62}b_{61}\cdots b_{52})_2 - 1023$）を利用して、log 関数を使わずに、その $\log_{10}$ 値を推測する。

q2-3.c に提供されているコードのひな形に従い、それを埋めて、log_float 関数と log_double 関数を完成せよ。

このプログラムを完成してコンパイルすると、以下のように実行することが可能になる。

```
[ya000836@purple99 ~/q2]$ gcc -Wall -std=c99 -o q2-3 q2-3.c
[ya000836@purple99 ~/q2]$ ./q2-3 127 128
1.806,2.107
[ya000836@purple99 ~/q2]$
```

図 5: 実行例 q2-3

q2-4：ポインタを介した構造体の操作（中級課題）

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、記憶されていなければ `no data` を表示するプログラムを作成する。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。

q2-4.c に提供されているコードのひな形に従って、それを埋めて、set 関数と search 関数を完成せよ。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。

このプログラムを完成してコンパイルすると、以下のように実行することが可能になる。

```
[ya000836@purple99 ~/q2]$ gcc -Wall -std=c99 -o q2-4 q2-4.c
[ya000836@purple99 ~/q2]$ ./q2-4
1611001 a1611001
1611002 b1611002
1611003 c1611003
1611004 d1611004
1611005 e1611005
1611004
1611006
1611001                          #入力はこの行まで
d1611004
no data
a1611001

[ya000836@purple99 ~/q2]$
```

図 6: 実行例 q2-4

q2-5：ポインタを介した異なる構造体の操作（上級課題）

問 q2-4 は同じ種類の学生情報を処理するが、異なる学生情報を処理できるように改造せよ。図 7 に示すように、学部生と大学院生は学籍番号、アカウント名などが共通であるが、学部生は類に所属し、大学院生は、専攻

に所属し、指導先生が異なる。プログラムの実行時に 学生の種類を type で区別し、適切な構造体を選択し、必要な領域を確保してから、学生の情報を蓄える。学生の種類の入力に応じて、指定された種類の学生の情報を id の昇順で出力せよ。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct student {
  int  type;              /* 学生種類 */
  int  id;                /* 学籍番号 */
  char account[9];        /* アカウント名 */
};
struct undergraduate {
  int  type;              /* 学部生：１０ */
  int  id;                /* 学籍番号 */
  char account[9];        /* アカウント名 */
  char program[9];        /* 学域名：I 類、II 類、III 類*/
};
struct graduate {
  int  type;              /* 大学院生：20 */
  int  id;                /* 学籍番号 */
  char account[9];        /* アカウント名 */
  char department[9];     /* 専攻名：I 専攻、J 専攻、S 専攻、M 専攻 */
  char supervisor[9];     /* 指導教員 */
};
```

図 7: 異なる種類の学生の構造体

q2-5.c に提供されているコードのひな形に従って、それを埋めて、set_undergraduate 関数、set_postgraduate 関数、search 関数などを完成させよ。

ヒント

- 学生種類に応じて、malloc を用いてプログラム実行中に構造体の領域を確保して、学籍番号、アカウント名などをセットしていく。学籍番号、アカウント名などの組を登録する部分は、空行等で登録を終了するようにする。
- 学生の情報を id に従って整列する際、構造体を移動せずに、指定された学生種類の構造体のポインタからなる配列を、id の昇順で整列することが望ましい。

```
[ya000836@purple99 ~/q2]$ gcc -Wall -std=c99 -o q2-5 q2-5.c
[ya000836@purple99 ~/q2]$ ./q2-5
10 1611001 a1611001 Field-I
20 1611006 f1611006 Depart-S Ito
20 1611004 d1611004 Depart-I Suzuki
10 1611003 c1611003 Field-III
20 1611005 e1611005 Depart-J Morigawa
10 1611002 b1611002 Field-II
10                                          #入力はこの行まで
10 1611001 a1611001 Field-I
10 1611002 b1611002 Field-II
10 1611003 c1611003 Field-III
[ya000836@purple99 ~/q2]$
```

図 8: 実行例 q2-5