2010 年度「プログラミング演習 2」 資料

立命館大学情報理工学部

1. 概要

本演習では「データ構造とアルゴリズム」の講義に対応させながら, Linux 環境で C 言語のプログラミング演習を行う. 演習内容は, 基本的なデータ型, ファイル入出力処理, データ構造およびアルゴリズムの実現方法を含んでいる.

前期に開講した「プログラミング演習1」で学んだ事を基礎にしつつ, 代表的なデータ構造(スタック, キュー, リスト, 二分探索木など)や基本的アルゴリズム(前記データ構造を実現するアルゴリズム, 各種ソートなど)のプログラムが作成できるようになることを目的とする.

また, **本演習は「プログラミング演習1」を修め, また「データ構造とアルゴリズム」の講義を受講していることを前提としている**.「プログラミング演習 1」の学習について不安があるものは必ず自学自習により復習を行い,「データ構造とアルゴリズム」については必ず演習当日分の内容までは, 学習した上で演習に望むこと.

2. 資料など

前期の「プログラミング言語」および「プログラミング演習1」で用いた下記の教科書を持参すること.

- C言語によるプログラミング　基礎編第２版 内田　智史：監修 オーム社
- 渡邉敏正著,「データ構造と基本アルゴリズム」, 共立出版, 東京, 2000.

また, 本演習の参考書として以下のものを挙げる.

- 平林雅英著,「ANSI C/C++辞典」, 共立出版, 東京, 1996.
- 島内剛一他著,「アルゴリズム辞典」, 共立出版, 東京, 1994.
- 近藤嘉雪著,「定本 Cプログラマのためのアルゴリズムとデータ構造」, ソフトバンククリエイティブ, 東京, 1998.
- 奥村晴彦著,「C言語による最新アルゴリズム事典」, 技術評論社, 東京, 1991.

3. 演習課題

各テーマについて, 必須課題とオプション課題を設定する. 必須課題は演習時間内にすべて完成させ, 時間に余裕があればオプション課題に取り組むこと. プログラムを完成できた時点で課題ごとに教員または TA･ES にプログラム動作の確認をしてもらうことで課題を終了したものと判断する. 必須課題に対する進捗の遅れは減点対象とする. 具体的には2週間以内にチェックを受けることを求める. それ以降のチェックについては減点対象とする. ただし, 14 週の必須課題については, 1 週の猶予しかないために注意すること. オプション課題については進捗の遅れに関わらず加点を行うので, 必須課題を優先して解くこと. また, 動作の確認時, 教員, TA･ES よりプログラムについて質問することがある. 質問に答えられ

なかった場合には，その課題については課題の終了とはしないので，各自でプログラムについて十分に理解しておくこと．また，プログラミングに関して不明な点は積極的に TA･ES に相談すること．

(注) フォローアップティーチングについて　毎週火曜の 11･12 限に P32 にて，金曜の 11･12 限に P31 にてフォローアップティーチングを実施する．フォローアップティーチングは進捗が遅れている学生に対してフォローアップを行う取り組みであり，時間内は TA が常駐する．進捗が遅れている学生は積極的に参加すること．**ただし，フォローアップティーチング中に課題完了の確認は行わない**．チェックは各クラスで受けること．

4. 成績評価

演習課題の到達度，小テスト，出席評価，演習態度などの日常点を総合的に評価する．また，原則として 10 回以上の出席，すべての必須課題の完成を単位取得の必要条件とする．なお，15 分以内の遅刻･早退は 1/3 欠席，45 分以内の遅刻･早退は 2/3 欠席，それ以上の遅刻･早退は欠席として扱う．**小テストは第6週（ポインタ，ファイル入出力など），第10週（リスト，スタック，キュー），第14週（ソート）において行う**ので必ず受験すること．

5. 注意事項

(1) 演習を行う前に必ず演習課題についての説明，及び「プログラミング言語」と「データ構造とアルゴリズム」の教科書に書かれた対応する内容を理解しておくこと．

(2) 出欠調査後の退出については，教員に届け出て許可をもらうこと．勝手に出て行った場合は欠席として扱うこともある．

(3) 演習時間内にプログラムが完成しない場合，演習時間外に作成し，次回の演習時間内に教員または TA･ES のチェックを受けること．

(4) 作成したプログラムについては，チェックの前に必ず自分で動作テストを実施し，動作を確認すること．動作しないプログラムについては，完成と認めない．

（5） 変数･定数･関数には分かりやすい名前を付け，プログラム中には適宜コメントを付ける習慣を付けること．また，各ファイルの先頭にはプログラムの説明と作成者，作成日を記載すること．

※ ハッシュタグについて

twitter などで利用する公式ハッシュタグを #prog2bkc と設定する．学生間の情報交換などで自由に利用すること．

第１週 ファイル入出力と構造体

▷ concept

ファイル入出力について学ぶ．また，構造体の復習を行い，これまで学んだものよりかは多少複雑な構造体の配列というデータ構造に触れる．

lecture keywords

FILE, fopen, fclose, fscanf, fprintf, ファイル終了判定 EOF, オープンモード ”r” ”w” ”a”, 構造体配列，構造体初期値設定，ユーザ定義型 typedef

√ 準備

下記の横線で囲まれた内容を ”gifts.dat”として作成し作業フォルダに保存せよ．この際には整形の為，スペースで区切ってもタブ文字’¥t’ で区切っても構わない．

<gifts.dat>

JZK-30	Jizake_ tsumeawase	4500
BSP-15	Body_ soap_ set	3000
BT-200	Bath_ towel_ set	2500
TEA-20	Koutya_ tsumeawase	5000

必須課題 1-1

ファイル gifts.dat を読み込み，その内容を標準出力に出力するプログラムを作成せよ. 読み込み時には, fscanf の書式指定子として %*s* もしくは %*d* をデータの種類に合わせて用いること．

必須課題 1-2

実行すれば, 下記の内容をファイル helloworld.txt として出力するプログラムを作成せよ．

```
Hello world !!
When I woke up this morning, I found many people in my room.
What’s up ?? ：-0
```

※ファイル入出力関数を用いて作成すること．リダイレクトによる入出力は認めない．

必須課題 1-3

code(文字列), name(文字列), price(整数) の３つのメンバ変数を持つ構造体 gift を定義せよ. gifts.dat の最初の一行と同じ内容を，宣言と同時にその初期値として格納せよ. その上で, 次の形式で構造体の内容を画面に出力するプログラムを作成せよ.

---

code: JZK-30
name: Jizake_ tsumeawase
price: 4500

※ただし，本問では構造体の定義には必ず typedef を用いてユーザ定義型として gift 型を設定すること.

必須課題 1-4

課題 1-1 で読み込んだ gifts.dat の内容を構造体 gift 型の配列に格納し，各商品名を下記の形式で順に表示するプログラムを作成せよ.

---

code: JZK-30
name: Jizake_tsumeawase
price: 4500

code: BSP-15
name: Body_ soap_ set
price: 3000
(以下略)

オプション課題 1-5

gifts.dat を読み込み，小文字を全て大文字に変換し，課題 1-4 に示したような形式でファイル出力するプログラムを作成せよ．また，gifts.dat に自由なデータを 6 行追加し動作することを確認せよ．また，この際に price が空白の行を設け，price が空白であった場合には

---

price: xxxx

と出力するようにせよ.

(ヒント)
空白に対応するためには fscanf の戻り値の利用, fgets の利用等が考えられる. また, ファイル終端の

判定には feof 関数を用いても良い. 小文字, 大文字の変換にはアスキーコードで加算減算する方法の他に toupper 関数や tolower 関数の利用が考えられる. これらについては自ら調べて用いること.

第 2 週 配列とポインタ

▷concept

ポインタを初めて学ぶ．ポインタについての概説とともに，配列の位置とポインタの関係について学ぶ．また，文字列と配列の関係について復習する．

lecture keywords

ポインタ，アドレス，変数とポインタ変数，リダイレクション, シングルクォーテーション (′), ダブルクォーテーション (“)

必須課題 2-1

以下のプログラムを入力し，コンパイルして実行せよ．実行結果を確認し，表示されたそれぞれの数値が何を意味しているかをプログラムの上にコメントとして記述せよ．

```
#include <stdio.h>

main()
{
        int a, *p; //(例) 整数型の変数 a　とポインタ変数 p を宣言している．

        a = 10;
        printf("a: %d\n", a);
        printf("&a: %p\n\n", &a);

        p = &a;
        printf("p: %p\n", p);
        printf("*p: %d\n", *p);
        printf("&p: %p\n\n", &p);

        a = 20;
        printf("a: %d\n", a);
        printf("&a: %p\n\n", &a);

        printf("p: %p\n", p);
        printf("*p: %d\n", *p);
        printf("&p: %p\n", &p);
}
```

必須課題 2-2

以下のプログラムを入力し，コンパイルして実行せよ．実行結果を確認し，表示されたそれぞれの数値が何を意味しているかプログラムの上にコメントとして記述せよ．

```
#include <stdio.h>

main()
{
        char c, *p;
        char s[12] = "Ritsumeikan";

        c = 'A';
        p = &c;
        printf("c: %c\n", c);
        printf("&c: %p\n\n", &c);
        printf("p: %p\n", p);
        printf("*p: %c\n\n", *p);

        *p = 'B';
        printf("c: %c\n", c);
        printf("&c: %p\n\n", &c);
        printf("p: %p\n", p);
        printf("*p: %c\n\n", *p);

        printf("s: %s\n", s);
        printf("s[0]: %c\n", s[0]);
        printf("s[1]: %c\n", s[1]);
        printf("s: %p\n", s);
        printf("&s[0]: %p\n", &s[0]);
        printf("*s: %c\n", *s);
        printf("*(s+1): %c\n\n", *(s+1));

        *(s+2) = 'T';
        printf("s: %s\n", s);
}
```

必須課題 2-3

下記のプログラムの下線部を埋めて，文字列を逆順に出力するプログラムを完成させよ．

```
#include <stdio.h>
int main( void ) {
  char *p1, *p2;
  p2 = “ Winter ”;    p1 = p2;
  while ( *p1 != '\0' ) {
     ____________________________   ;    //  文字列の最後を検索
  }

  while ( ___________________ ) {        // アドレスを比較
     ____________________________   ;
     putchar( *p1 );   //アドレスを制御して文字を１つずつ出力
  }
  putchar( '\n' );
  return 0;
}
```

実行結果： retniW

オプション課題 2-4

英語の文章を EOF が現れるまで繰り返し標準入力から読み込み, 文字数（空白，コンマ・ピリオドなどの記号を含む）と「th」または「Th」の出現回数を出力せよ．また，テキスト中の「th」または「Th」をすべて「++」で置き換えたものを出力せよ．ただし，改行文字は文字数に含まない．実行例を以下に示すが，必ずしもこの通りの表示でなくてもよい．実行にはリダイレクションを用いても構わない．

実行例：

英文を入力して下さい.
We have seen other things matter.
But, that is the thing that matters.
^D
We have seen o++er ++ings matter.
But, ++at is ++e ++ing ++at matters.
文字数：69
th の出現回数：6

※「^D」は EOF のキーボードからの入力を意味し,「Ctrl」キーを押しながら「D」キーを押すことで入力される．

第３週 動的メモリ確保とポインタ配列

▷ concept

動的メモリ確保について学び, ポインタと配列, 文字列の関係について理解を深める. また, ポインタを配列状に確保し, それらが別のメモリ領域を指すというデータ構造について学ぶ.

lecture keywords

malloc, free, size_t, キャスト演算子, ポインタ配列

必須課題 3-1

C 言語では文字列を char 型配列で扱おうとすると，コンパイル時に文字列の最大の長さを決めなければならなくて不便に思われる．そこで，malloc を用いた動的メモリ確保により，この問題を解決したい．文字列の長さを入力させた後に，任意の長さの文字列を標準入力から受けつけ，読み取るプログラムを作成せよ．また，読み込んだ文字列を大文字にした上で，順序を逆にして出力するプログラムを作成せよ.

動作例:

```
文字列最大長を入力してください：26
文字列を入れて下さい: abcdefghijklmnopqrstuvwxyz
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

必須課題 3-2

最大５人の名前を格納するポインタ配列（例：char *name[5]）を宣言する．キーボードから１人ずつ名前を入力し，その名前の文字列を格納できるメモリ領域を確保してから名前の文字列を格納し，さらにそのメモリ領域へのポインタを配列に格納する．最後に，入力された人数分について，名前が格納されているアドレスと名前の文字列を出力するプログラムを作成せよ.

※ １人の名前の最大の長さは 20 バイト（全角文字で 10 文字分）とすること.
※ アドレスの出力では，printf() における書式指定で %*x* を用い 16 進数で表記すること.
※ メモリ領域へのポインタを配列に格納してから文字列を格納しても構わない.

実行例：(必ずしも実行例の通りでなくてもよい)

名前を最大 5 人分入力して下さい.
KusatsuTaro
KinugasaHanako
BeppuJiro
^D
0x8049810 KusatsuTaro
0x8049820 KinugasaHanako
0x8049830 BeppuJiro

必須課題 3-3

構造体のポインタ配列を定義し，課題 1 で作った gifts.dat をファイルから，構造体を利用する分だけ動的にメモリ確保し，データを読み取ることのできるプログラムを作成せよ．読み込んだ結果は，課題 1-4 と同様の形式でファイル出力せよ．また, 入力ファイル名を標準入力から指定できるようにせよ.

実行例：

データ数を入力して下さい：4
ファイル名を指定して下さい：gifts.dat
フォーマット変換後ファイルに出力しました.

第４週　ポインタ引数と文字列処理

▷ concept

値引数とポインタ引数の挙動の違いを理解する．基本的な文字列操作関数を学び, その扱いを覚える．

lecture keywords

値渡し，ポインタ渡し，strlen, strcpy, strcat, strcmp, ⟨string.h⟩, 関数のプロトタイプ宣言，ポインタによる文字列渡し

必須課題 4-1

変数 x, y の値を入れ替える関数を作りたい．以下のプログラムを作ったところ，上手く行かなかった．

```
#include <stdio.h>

void swap(int x, int y);

int main(){
  int x,y;
  x = 2; y = 5;
  swap(x,y);
  printf("x=%d, y=%d\n",x,y);
}

void swap(int x, int y){
  int temp = x;
  x = y;
  y = temp;
}
```

このプログラムを入力し，出力結果を観察せよ．また，なぜ上手くいかないのか口頭で説明せよ．

必須課題 4-2

引数で *x* と *y* を取得し, この値を入れ替えるためには,「ポインタ引数」で関数に渡せばいいと考えられる．上記，swap 関数を改変し，x,y の値を入れ替えられるようにせよ．これを用いて，２つの整数値をキーボードから入力してそれぞれ変数に代入し，２つの変数の値を入れ換えて出力するプログラムを作成せよ.

実行例：

```
２つの整数値を入力して下さい：3 9
変数 x の値は 3，変数 y の値は 9 です.
swap() を呼び出した後の変数 x の値は 9，変数 y の値は 3 です.
```

必須課題 4-3

２つの文字列を保存する構造体 word_pair を下記の様に定義する.

```
typedef struct word_pair{
        char longer_word[10];     //長い方の文字列
        char shorter_word[10];    //短い方の文字列
        char combined_word[21];   //連結した文字列
        int longer_word_length;   //長い方の文字列の長さ
        int shorter_word_length;   //短い方の文字列の長さ
} word_pair_t;
```

この構造体を新たに作成し, データをセットして返す関数
word_pair_t create_word_pair(char *a, char *b);
を作成せよ．create_word_pair は以下の仕様を満たす.

create_word_pair は２つの文字列 a と b を比較し，長い文字列を longer_word に，短い文字列を shorter_word に代入する．また，これらの長さが同じ場合には辞書的に後ろのものを longer_word に，前のものを shorter_word に代入する．もし，a と b がまったく同じ文字列であれば，エラーメッセージを出力した上で，longer_word に入力された文字列を，shorter_word に空の文字列 (もしくは先頭がヌルの文字列) を代入する．また，combined_word には longer_word と shorter_word をスペース区切りで結合したものを代入する．さらに，それぞれの文字列の長さを longer_word_length, shorter_word_length に代入する.

標準入力から文字列を２つ読み取り，create_word_pair を用いて，新たにそれらのデータが代入された構造体を作成したのちに，これらのメンバ変数を全て，標準出力に表示するプログラムを作成せよ.

※ このプログラムには string.h に定義された種々の関数を積極的に用いること.

オプション課題 4-4

「⟨ 数字列 ⟩⟨ 四則演算子 ⟩⟨ 数字列 ⟩」の形をした 2 項演算の数式を文字列としてキーボードから入力し，四則演算子を区切り記号として文字列を 3 つに分割し，プログラム内部にて配列に格納した後に出力するプログラムを作成せよ．ここで ⟨ 四則演算子 ⟩ は「+」「-」「*」「/」のいずれかであり，⟨ 数字列 ⟩ は数字（「0」～「9」）の 1 個以上の並びである．さらに，入力された数式の計算結果を表示するプログラムを作成せよ．また,「⟨ 数字列 ⟩⟨ 四則演算子 ⟩⟨ 数字列 ⟩」の形式ではない文字列が入力された場合は「ERROR」という文字列を出力するようにせよ．なお，整数を表す文字列を整数値に変換するために，標準ライブラリ関数 atoi() を用いることができる．

実行例：

```
数式を入力して下さい：12+345
数値 1：12
演算子：+
数値 2：345
計算結果：357
```

実行例 2（エラーの場合）：

```
数式を入力して下さい：67%8
  ERROR
```

第５週 配列を使ったスタック

▷ concept

スタックの基本的な概念と配列を使ったスタックの表現について学ぶ.

lecture keywords

スタック，push, pop，配列によるスタックの表現，#define

1 次元配列を用いてスタックを実現する．スタックとは次のような操作を可能とするメモリの一種である.

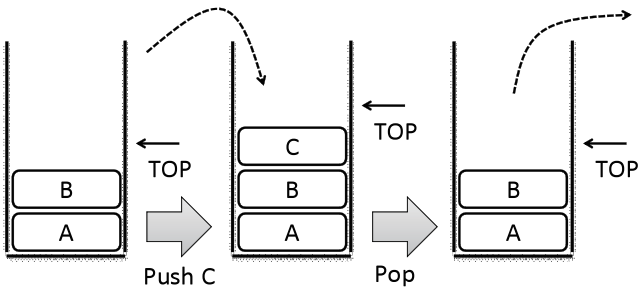

図 1: スタック

Push (data) : data を push-down する.（データをスタックに格納する.）

Pop: data を pop-up する．(最も新しく格納したデータを読み出す．)

必須課題 5-1

配列を用いてスタックを表現する．その際，配列の添字 0 の要素をスタックの底とし，データが入っている要素の最大の添字より１つ大きな値を top という変数で保持することでスタックのデータ構造を表現する．この時，スタックの内容を出力する関数
void print_stack_ary(char* s, int top);
を作成せよ．ここでスタックのデータは char 型の配列により保存するものとする.

出力例 : スタックの中身が [h,o,g,e] で top が 4 の場合

```
<---TOP=4
e
g
o
h
```

必須課題 5-2

1 次元配列 s[MAX] と変数 top を用いてスタックの基本操作を行うプログラムを実現したい．まず，スタックに文字を push-down する関数
void push(char c, char* s, int* top);
を作成せよ．これが正しく動作することを push-down 動作の前と後, それぞれ内容を print_stack_ary を用いて出力する事で確認せよ．ここで，MAX は配列の大きさを与える定数である．

(ヒント) top の値は push 関数内で 1 増えることになる．

必須課題 5-3

次に，スタックから 1 文字を pop-up する関数
char pop(char* s, int* top);
を作成せよ．print_stack_ary によってスタックの中身を出力した後に pop を行い，pop した文字を出力せよ. その後に，再度 print_stack_ary を行い，データが 1 つ減っている事を確認せよ．

オプション課題 5-4

キーボードからの push-down / pop-up 機能を追加する．下記の仕様を満たすようにプログラムを作成せよ．

1. キーボードからキー（文字）を 1 文字ずつ入力して，スタックを操作する．
2. スタック操作の仕様はキーボードからキー（文字）を 1 文字ずつ入力する際に，
    - 0 を入力した場合，プログラムを終了
    - 1 を入力した場合，1 文字 pop-up した後， pop-up した文字とスタックの内容を表示
    - その他の文字を入力した場合，その文字を push-down した後，スタックの内容を表示

- スタックの内容の表示は print_stack_ary を使用すること

オプション課題 5-5

必須課題 5-4 を拡張して 0,1 以外の文字が入力された場合，そのまま push-down するのではなく，その文字がスタック中にあるか否かを探索し，スタック中にあれば「その文字をスタックから削除」，なければ「その文字をスタックに追加」するプログラムを実現せよ．ここでは，図 2 のようにスタックを 2 個用意（状態 A）して，Stack1 から 1 文字ずつ pop-up してスタック内の探索を行い，文字を削除するか Stack2 に文字を push-down するか判断（状態 B）する．その後，Stack2 から文字を 1 文字ずつ pop-up して Stack1 に push-down（状態 C）し，最後に入力文字をスタックに追加する必要があれば入力文字を Stack1 に push-down すること．ただし，2 つのスタックに対する push-down と pop-up はそれぞれ 1 つの関数で行い，引数により各スタックへの操作を切り替えること．

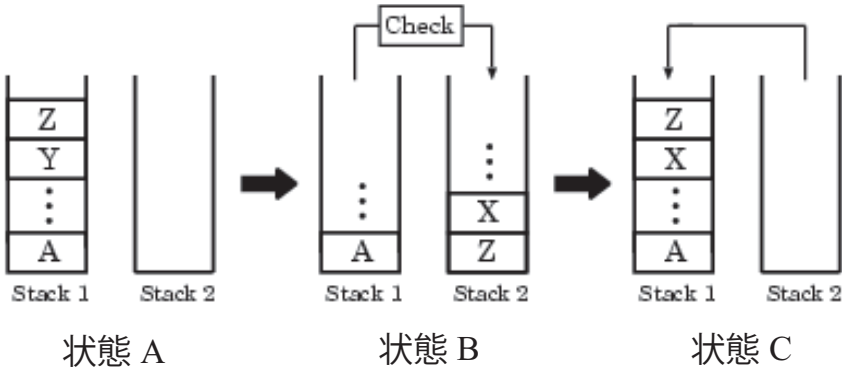

図 2: スタックの探索

第６週 配列を使ったキュー

▷ concept

配列を使ったキューの表現について学ぶ.

lecture keywords

キュー，enqueue, dequeue，配列によるキューの表現

1 次元配列を用いてキューを実現する.
キューとは次のような操作を可能とするメモリの一種である.

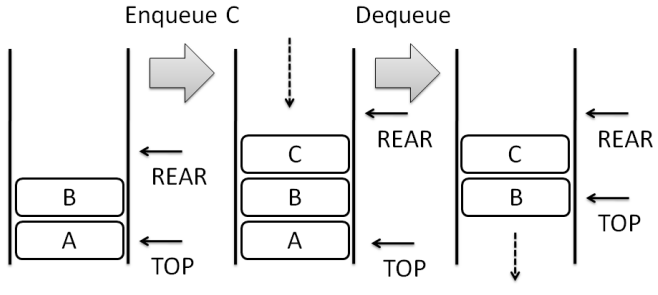

図 3: キュー

Enqueue (data)： データをキューに格納する.

Dequeue： 最も古く格納したデータを読み出す.

この週で扱うキューでは，データが配列の末尾を超えない範囲を扱うものとするので，データが配列の末尾を超えた際の処理は考慮しなくて良い.

必須課題 6-1

配列を用いてキューを表現する．その際，配列の添字 top の要素をキューの先頭とし，データが入っている要素の最大の添字より１つ大きな値を rear という変数で保持することでキューのデータ構造を表現する．この時，キューの内容を出力する関数
void print_queue_ary(char* q, int top, int rear)
を作成せよ.

出力例 : キューの中身が [h,o,g,e] で top が 3, rear が 7 の場合

```
h<---TOP=3
o
g
e
<---REAR=7
```

必須課題 6-2

1 次元配列 s[MAX] と変数 top, rear を用いてキューの基本操作を行うプログラムを実現したい．まず，スタックに文字を enqueue する関数
void enqueue(char c, char* q, int* top, int* rear)
を作成せよ．
これが正しく動作することを enqueue 動作の前と後の，それぞれのキューの内容を
print_queue_ary を用いて出力する事で確認せよ．ここで MAX 配列の大きさを与える定数である．

必須課題 6-3

1 次元配列 s[MAX] と変数 top, rear を用いてキューの基本操作を行うプログラムを実現したい．次に，キューから 1 文字を dequeue する関数
char dequeue(char* q, int* top, int* rear)
を作成せよ．

print_queue_ary によってキューの中身を出力した後に dequeue を行い，dequeue した文字を出力した後に，再度 print_queue_ary を行い，データが 1 つ減っている事を確認せよ．

オプション課題 6-4

キーボードからの enqueue / dequeue 機能を追加する．下記の仕様を満たすようにプログラムを作成せよ．

1. キーボードからキー（文字）を 1 文字ずつ入力して，キューを操作する．
2. キュー操作の仕様はキーボードからキー（文字）を 1 文字ずつ入力する際に，
    - 0 を入力した場合，プログラムを終了
    - 1 を入力した場合，1 文字 dequeue した後，dequeue した文字とキューの内容を表示

- その他の文字を入力した場合，その文字を enqueue した後，キューの内容を表示
- キューの内容の表示は print_queue_ary を使用すること

オプション課題 6-5

課題 6-4 を拡張して 0,1 以外の文字が入力された場合，そのまま enqueue するのではなく，その文字がキュー中にあるか否かを探索し，キュー中にあれば「その文字をキューから削除」，なければ「その文字をキューに追加」するプログラムを実現せよ．ここでは，図 4 のようにキューを 2 個用意（状態 A）して，Queue1 から 1 文字ずつ dequeue してキュー内の探索を行い，文字を削除するか Queue2 に文字を enqueue する（状態 B）．その後，Queue2 から文字を 1 文字ずつ dequeue して Queue1 に enqueue（状態 C）し，最後に入力文字をキューに追加する必要があれば入力文字を Queue1 に enqueue すること．ただし，2 つのキューに対する enqueue と dequeue はそれぞれ 1 つの関数で行い，引数により各キューへの操作を切り替えること．

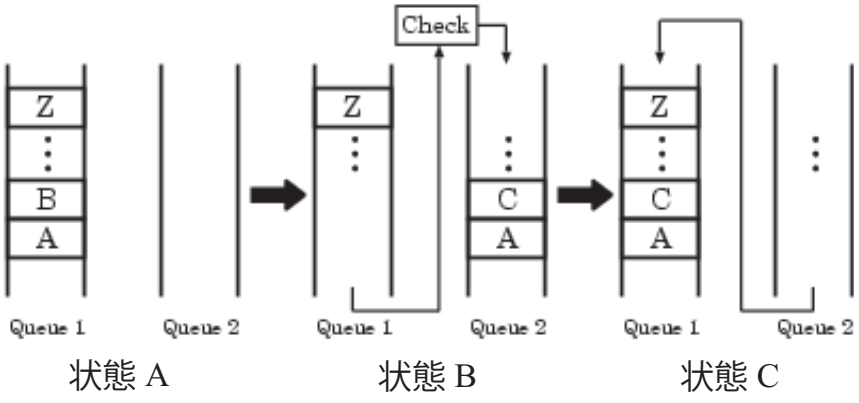

図 4: キューの探索

第７週 リスト

▷ concept

構造体を繋ぎ合わせる事で実現するリストの概念を理解し実装する．

lecture keywords

単方向線状リスト，メンバであるポインタ変数への malloc

下のように data 型の構造体を定義

```
struct data{
        char key;
        struct data *next;
};
```

必須課題 7-1

data 型構造体のポインタ変数 top を作成しリストヘッドとする. 下記の出力命令に対し下記の出力結果を得るように，malloc を用いてメモリ確保を行ない，リスト末尾に要素を挿入していくことで，単方向線状リストを作成せよ．

〈出力命令〉

```
printf("%c\n",top->key);
printf("%c\n",top->next->key);
printf("%c\n",top->next->next->key);
```

〈出力結果〉

```
a
b
c
```

この時，リストは下記のような形になっている．

※ 図の見方はデータ構造とアルゴリズムの教科書参照

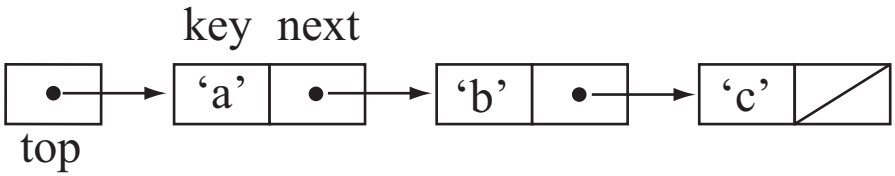

図 5: 単方向線状リスト

必須課題 7-2

課題 7-1 で作成したリストの key を順に表示する関数 void print_stack_list(struct data *top) を作成せよ．出力の形式は課題 5-1 と同様に，下記のようになるようにせよ．また，リストへの挿入時，新たな要素はリストの先頭に挿入するように変更せよ．ここで，このリストは第８週で扱うスタックを表現するリストとなっている．

出力例 : スタックの中身が [h,o,g,e] の場合

```
e<---TOP
g
o
h
```

必須課題 7-3

始めのポインタだけでなく，末端のポインタも持つ単方向線状リストを作成する．
下記のように queue 型の構造体を定義し，リストヘッドとする．

```
struct queue{
        struct data *top, *rear;
};
```

queue 型の構造体 q を作り，これを用いて，下図で表されるリストを作成し

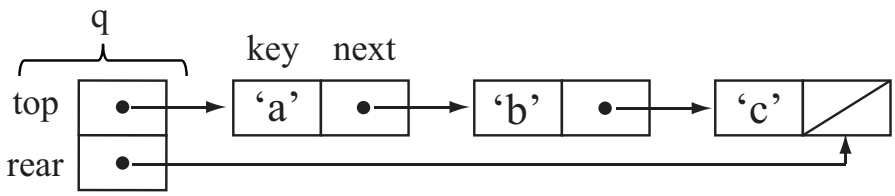

図 6: 単方向線状リスト (rear のある場合)

以下の出力命令に対して，以下の出力結果が得られるようにせよ．

〈出力命令〉

```
printf("%c\n",q.top->key);
printf("%c\n",q.top->next->key);
printf("%c\n",q.rear->key);
```

〈出力結果〉

```
a
b
c
```

必須課題 7-4

課題 7-3 で作成したリストの key を順に表示する関数
void print_queue_list(struct queue q)
を作成せよ．
出力の形式は課題 6-1 と同様に下記のようになるようにせよ．
ここで，このリストは第８週で扱うキューを表現するリストとなっている．

出力例 : キューの中身が [h,o,g,e] の場合

```
h<---TOP
o
g
e<---REAR
```

第８週　リストを用いたスタックとキュー

▷ concept

リストを用いてスタックとキューを実装する．リストの挿入や削除の概念を学ぶ．また，それを通して物理メモリに縛られないポインタによる論理的データ結合を学ぶ.

lecture keywords

リストの結合とポインタの代入の関係, ポインタのポインタ, リストを用いたスタックでの push と pop

必須課題 8-1

第７週 課題 7-2 で作成したリストで表現されたスタックに対して，push を行う関数を作成せよ．push 関数を呼び出す前と後で print_stack_list を呼び出し，push 関数が正常に働いていることを確認せよ．

(ヒント) push 関数のプロトタイプ宣言は下のようにできる．

```
void push (struct data **top, char key);
```

必須課題 8-2

第７週 課題 7-2 で作成したリストで表現されたスタックに対して，pop を行う関数を作成せよ．pop 関数を呼び出す前と後で print_stack_list を呼び出し，pop 関数が正常に働いていることを確認せ.
(ヒント) pop 関数のプロトタイプ宣言は下のようにできる．

```
char pop (struct data **top);
```

必須課題 8-3

第７週 課題 7-4 で作成したリストで表現されたキューに対して，enqueue を行う関数を作成せよ．enqueue 関数を呼び出す前と後で print_queue_list を呼び出し，enqueue 関数が正常に働いていることを確認せよ．

(ヒント) enqueue 関数のプロトタイプ宣言は下のようにできる．

```
void enqueue (struct queue *q, char key);
```

必須課題 8-4

第７週で作成したリストで表現されたキューに対して，dequeue を行う関数を作成せよ．dequeue 関数を呼び出す前後で print_queue_list を呼び出し，dequeue 関数が正常に働いていることを確認せよ．
(ヒント) dequeue 関数のプロトタイプ宣言は下のようにできる．

```
char dequeue (struct queue *q);
```

オプション課題 8-5

リストで表現されたキューにキーボードからの enqueue / dequeue 機能を追加する．下記の仕様を満たすようにプログラムを作成せよ．

1. キー（文字）を読み込みキューに enqueue し，キューの内容を表示した後に，キーボードからキー（文字）を 1 文字ずつ入力して，キューを操作する．
2. キュー操作の仕様はキーボードからキー（文字）を 1 文字ずつ入力する際に，
   - 0 を入力した場合，プログラムを終了
   - 1 を入力した場合，1 文字 dequeue した後，dequeue した文字とキューの内容を表示
   - その他の文字を入力した場合，その文字を enqueue した後，キューの内容を表示
   - キューの内容の表示は print_queue_list を使用

   すること．

オプション課題 8-6

リストで表現されたスタックにキーボードからの push-down / pop-up 機能を追加する．下記の仕様を満たすようにプログラムを作成せよ．

1. キーボードからキー（文字）を 1 文字ずつ入力して，スタックを操作する．
2. スタック操作の仕様はキーボードからキー（文字）を 1 文字ずつ入力する際に，
   - 0 を入力した場合，プログラムを終了
   - 1 を入力した場合，1 文字 pop-up した後，pop-up した文字とスタックの内容を表示
   - その他の文字を入力した場合，その文字を push-down した後，スタックの内容を表示
   - スタックの内容の表示は print_stack_list を使用

   すること．

第９週 ソート（１）

▷ concept

選択ソートと挿入ソート, バブルソートについて理解し，簡単な配列に対して実行できるようになる．

lecture keywords

挿入ソート，選択ソート，バブルソート

√準備

下記の内容のファイルを”numbers.dat”として作成せよ．

numbers.dat

```
91 63 71 14 60 1 24 13 80 15
```

必須課題 9-1

numbers.dat から 10 個の整数を読み込み，選択法でソートし，小さい順に表示せよ．

必須課題 9-2

numbers.dat から 10 個の整数を読み込み，挿入法でソートし，小さい順に表示せよ．

オプション課題 9-3

numbers.dat から 10 個の整数を読み込みバブルソートでソートし，小さい順に表示せよ．

オプション課題 9-4

　データフォルダに国別の国名と地域，人口を記入したファイル”countries16.dat”がある．
これを構造体配列に読み込み，人口を降順にソートするプログラムを作成し，ソート結果を出力せよ．ソートの方法は任意とする．
(データフォルダ: http://www.em.ci.ritsumei.ac.jp/prog2)

第１０週　ソート（２）

▷ **concept**

ヒープ構造を理解することで，ヒープソートの準備とする．

lecture keywords

ヒープ，ヒープの 1 次元配列表現, insert, deletemin

必須課題 10-1

numbers.dat に含まれるデータをヒープ条件が満たされるように，ノートもしくは余白の上に木構造で書き表せ．
このヒープを一次元配列表現し，プログラム中の配列に初期値として設定し標準出力へ出力せよ．この出力された数字の並びが正しいかどうかについてチェックを受けよ. ここで，構成するヒープは「ノードのキーはその親のキーよりも大きいか等しい」という最小ヒープの性質を満たすものとする．

必須課題 10-2

一次元配列として表現されたヒープ a がヒープ条件を満たしているか確かめる関数
int check_heap(int *a, int N)
を作成せよ，ヒープ条件を満たしていれば 1, 満たしていなければ 0 が返値となる．N は一次元配列内でヒープのメモリとして利用されている要素の数である．
また，これを課題 10-1 のデータに対して適用し，ヒープ条件を満たしていることを確認せよ．

必須課題 10-3

ヒープに新たなデータを挿入する関数 insert を作成せよ. 空の配列に複数回 insert を行うことで構成したヒープがヒープ条件を満たしていることを check_heap を用いることで確認せよ. プロトタイプ宣言は以下とする．

```
void insert(int val, int a[], int N);
```

必須課題 10-4

ヒープの最小値を取り除く，deletemin 関数を作成せよ．空の配列に課題 10-3 で作成した insert を複数回実行することでヒープ条件を満たすヒープを構成する. これに deletemin を実行することで最小値を抽出せよ. 最小値を抽出した後，一次元配列がヒープ条件を満たすように再構成せねばならない. プロトタイプ宣言は以下とする.

```
int deletemin(int a[], int N);
```

第１１週　ソート（３）

▷ **concept**

ヒープソート，マージソート，クイックソートを実現する．

lecture keywords

ヒープソート，マージソート，クイックソート

必須課題 11-1

numbers.dat から 10 個の整数を読み込みヒープソートでソートし，小さい順に表示せよ．

オプション課題 11-2

numbers.dat から 10 個の整数を読み込みマージソートでソートし，小さい順に表示せよ．

オプション課題 11-3

numbers.dat から 10 個の整数を読み込みクイックソートでソートし，小さい順に表示せよ．

オプション課題 11-4

上記の numbers.dat 以外に，データフォルダにはデータ量を変えたファイル

numbers1K.dat (データ数: 1,000)

numbers10K.dat (データ数: 10,000)

numbers100K.dat (データ数: 100,000)

がある．このファイルには 1 行目にデータ数, 2 行目以降に 1 行ずつデータが記述されている．これらを対象にそれぞれ選択法，挿入法，ヒープソート，クイックソートでソートを行い，処理時間を比較せよ．プログラム内で関数などの処理時間を測定するには，関数を実行する前後で gettimeofday() などで時刻に関する情報を取得し，その差をみればよい．
gettimeofday() の使い方については
man gettimeofday
等を参照のこと．
(データフォルダ: http://www.em.ci.ritsumei.ac.jp/prog2)

第１２週　木構造と探索（１）

▷ concept

木構造の基本と二分探索木について学ぶ．構造体とポインタを用いた二分探索木の実現について学び，単方向線状リストとの関係を意識しながら二分探索木の実装を行う．また，外部から与えられた関数を分割コンパイルを用いることで利用することを学ぶ．

lecture keywords

二分探索木，分割コンパイル，正則二分木，外点

二分探索木を扱う．各要素を表現する構造体 struct node 型 は下記で定義される．

```
struct node {
    int key;
    struct node *parent, *left, *right;
};
```

print_tree() については，print_tree.c に定義されている．以下のプログラムでは, 作成するプログラムの main 関数より上で print_tree のプロトタイプ宣言を行うことで，分割コンパイルを用いてこれを利用せよ．print_tree.c は, データフォルダ: http://www.em.ci.ritsumei.ac.jp/prog2 に置いてある．

※ 分割コンパイルについて
prog.c と func.c にプログラムが分かれている場合.
gcc -c prog.c を実行すると prog.o が作成される. 同様に
gcc -c func.c を実行すると func.o が作成される.
この２つのオブジェクトに対して
gcc prog.o func.o -o prog とすると実行ファイル prog を作成することができる.

必須課題 12-1

上の構造体を用いて，教科書 p.154 図 6.1 に示される 2 分探索木を作成せよ．その際，新しい点の構造体を作成する関数 new_node() を作成して使用すること．なお，ここでは，すべての点の構造体を作成してからそれらのポインタを手動で繋ぎ合わせること．ただし，正則 2 分木になるように外点も作成すること（教科書の図 6.3 参照）．作成後，木の構造を print_tree() 画面に表示せよ．また，関数 new_node() および print_tree() の仕様は以下の通りとする．

struct node *new_node(int key)

・機能:　構造体 node 型の領域を確保し，要素の値を代入する．親・子を指すポインタはすべて NULL で初期化する．

・引数 key:　要素の値 (キー)

・戻り値:　作成した構造体 node 型の要素へのポインタ

void print_tree(struct node *node)

・機能:　引数 node を始点とする 2 分探索木の構造を画面に表示する．
ただし，画面の左端を根として右側に枝が伸びる形で表示されることに注意．

・引数 key:　表示する 2 分探索木の始点

・戻り値:　なし

〈スケルトンコード〉

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

struct node {
    int key;
    struct node *parent, *left, *right;
};

struct node *new_node(int key)
{
    struct node *p;
/*  この間に適切なコードを書く */

    return p; //ただし p は 当該関数内で定義された struct node 型のポインタ
}

main()
{
    struct node *root, *n[10];

    n[0] = new_node(4);

/* この間に適切なコードを書く */

    n[5] = new_node(18);
```

```
/* この間に適切なコードを書く */

    root = n[5];

    n[0]->left = new_node(0);
    n[0]->right = n[1];
    n[0]->parent = n[2];

/* この間に適切なコードを書く */

    n[5]->left = n[2];
    n[5]->right = n[7];
    n[5]->parent = NULL;

/* この間に適切なコードを書く */

    print_tree(root);
}
```

実行例:

```
            0
        31
                0
            25
                0
    22
             0
        21
             0
  18
                0
            15
                0
        12
             0
     9
                0
             7
                0
         4
             0
```

第１３週　木構造と探索（２）

二分探索木における挿入と探索について理解し実装を行う．

▷ concept

lecture keywords

insert, member, 再帰

必須課題 13-1

キーボードから整数値を繰り返し入力し，それを要素として 2 分探索木の適切な位置に挿入するプログラムを作成せよ．その際，要素を挿入する関数 insert() を作成して使用すること．また，要素を挿入するごとに木の構造を画面に表示せよ正でない整数が入力された時点でプログラムを終了するようにせよ．また，すでに同じ値の要素がある場合には，エラーを表示するようにせよ．
関数 insert() の仕様は以下の通りとする．

int insert(int key, struct node *root)

・機能:　　2 分探索木を探索し，適切な位置に要素 key を挿入する．

・引数 key:　　挿入する要素の値（キー）

・引数 root:　　2 分探索木の根を示すポインタ

・戻り値:　　要素 key を挿入できた場合は 1，挿入できなかった場合（すでに同じ値の要素がある場合）は 0 を返す

実行例:

```
挿入するキーを入力して下さい: 18
      0
18
      0

挿入するキーを入力して下さい: 9
      0
18
          0
      9
          0
```

(中略)

```
挿入するキーを入力して下さい: 25
                    0
          31
                    0
              25
                    0
     22
               0
          21
               0
 18
                    0
              15
                    0

          12
               0
       9
                    0
               7
                    0
           4
               0
挿入するキーを入力して下さい: -1
```

オプション課題 13-2

課題 12-1 で作成したプログラムを拡張し，キーボードから入力した整数値を 2 分探索木から探索するプログラムを作成せよ．その際，再帰的に探索を行う関数 member_recursive() を作成して使用すること．再帰を用いた探索の一般的なアルゴリズムは以下の通りである．

まず，v を根 root に設定して以下の操作を行う．

Step 1 . v が外点ならば, “ キー x はありません．”と出力して終了する．

Step 2 ．（v は内点である）key(v) = x ならば“ キー x が見つかりました．”と出力して終了する．

Step 3 ．（key(v) ≠ x である）もし key(v) > x ならば v ← left(v).
もし key(v) > x ならば v ← right(v).

Step 1〜3 を実現する関数を再帰的に呼び出す.

関数 member_recursive() の仕様は以下の通りとする.

struct node *member_recursive(int key, struct node *node)

・機能: 2 分探索木を再帰的に探索し，その結果を返す.

・引数 key: 探索する要素の値（キー）

・引数 root: 2 分探索木の点を示すポインタ

・戻り値: 要素 key が見つかればその点を示すポインタを，見つからなければ NULL を返す

実行例 1：

```
探索するキーを入力して下さい：1
キー 1 はありません.
```

実行例 2：

```
探索するキーを入力して下さい：31
キー 31 が見つかりました
```

オプション課題 13-3

課題 12-1 で作成した 2 分探索木に対して，キーボードから整数値を入力して該当する要素を削除するプログラムを作成せよ．その際，要素を削除する関数 delete() を作成して使用すること．また，要素を削除するごとに木の構造を画面に表示せよ．正でない整数が入力された時点でプログラムを終了するようにせよ.

関数 delete() の仕様は以下の通りとする.

int delete(int key, struct node *root)

・機能: 2 分探索木をから引数 key で指定された要素を削除する.

・引数 key: 削除する要素の値（キー）

・引数 root: 2 分探索木の根を示すポインタ

・戻り値: 要素 key を削除できた場合は 1，削除できなかった場合（該当する要素がない場合）は 0 を返す

実行例:

```
削除するキーを入力して下さい: 12
              0
         31
                   0
              25
                   0
     22
              0
         21
              0
 18
              0
         15
              0
      9
                   0
              7
                   0
          4
              0
削除するキーを入力して下さい: 4
              0
         31
                   0
              25
                   0
     22
              0
         21
              0
 18
              0
         15
              0
      9
              0
          7
              0
削除するキーを入力して下さい: -1
```

第１４週　木構造と探索（３）

▷ **concept**

ハッシュ法について理解し，剰余を用いた基本的なハッシュを構成できる．

lecture keywords

ハッシュ，ハッシング，リスト集合，ポインタ配列，ポインタのポインタ

必須課題 14-1

整数 x を整数 y で割った剰余 (x mod y) を関数値とするハッシュ関数によりハッシュを構成せよ．ハッシュの構成方法としてはチェイン法を採用し，同一ハッシュ値をもつ要素を線状リストで管理する．具体的には y=5 として，教科書 p.179 図 6.48 に示されるリスト集合をプログラム上で構成せよ．リストヘッドにはポインタ型の１次元配列 struct node *table[5] を利用すればよい．このハッシュの内容を見やすく表示する関数 print_hash() を作成し，構成したハッシュを標準出力上に表示せよ．
各ノードは

```
struct node{
        int key;
        struct node *next;
}
```

で与えられるものとする．

必須課題 14-2

このハッシュに対して，ハッシュが 変数 x を含むかどうかを探索する関数

struct node *member(struct node **table, int key)

を構成し，正しく動作することを標準入出力をとおして確認せよ．

オプション課題 14-3

課題 14-1 で構成したハッシュに対して新たなデータを追加する関数 insert() を構築せよ．

オプション課題 14-4

課題 14-1 で構成したハッシュに対して該当するデータを削除する関数 delete() を構築せよ．

オプション課題 14-5

人名を保存するハッシュを作成せよ．ハッシュ関数として何を用いるかも自ら考えよ．このハッシュに対して member(), insert(), delete() 関数を作成し，一通りの操作ができるようにせよ．

第１５週　まとめ

▷ **concept**

データ構造とアルゴリズムでの学習内容をＣ言語で構築することについての理解を確認する．

lecture keywords

まとめ

※この週のオプション課題はこれまでの週の全てのオプション課題が終わった者のみ取り組むこと．また，チェックは教員が直接行う．課題における，幅優先探索，深さ優先探索については，データ構造とアルゴリズムでの講義及び教科書の内容を参考にして実装すること．

オプション課題 15-1

幅優先探索を行うプログラムを作成せよ．詳細な仕様は自ら決定すること．

オプション課題 15-2

深さ優先探索を行うプログラムを作成せよ．詳細な仕様は自ら決定すること．

オプション課題 15-3

これまでに学んだ知識と技術を用いて，有用性のある何らかのプログラムを構築せよ．詳細な仕様は自ら決定すること．※提出方法については各担当教員に問い合わせること．