C++プログラミング　課題編

第１章から第６章まではベターC としての C++の課題
第７章は Linux のシステムコール　fork/wit/exit/exec 及びメモリマップに関する課題
第８章は C++のファイル処理
第９章から 11 章まではオブジェクト指向に関わる課題

確認した動作環境は Linx（CentOS7)、gnuC++コンパイラ。

課題には難易度を付した。
印なし：初級
　「やさしい C++」などの参考書をひもとき文法を確認しながら作成する
＊：　中級　　簡単なアルゴリズムを考えながらプログラミングする
＊＊：上級　　業務プログラム（基本ソフトウェア）レベルの練習

課題作成にあたって参考にした本
１．高橋麻奈　「やさしい C++　第３版」　ソフトバンク　2009 年
２．柴田望洋　「解きながら学ぶ　C 言語」　ソフトバンク　2004 年
３．羽山博監修　垰井正雄著　「Linux システムプログラミング」
　　オーム社　平成 16 年
４．浅海智晴　「やさしい UML 入門」ピアソン・エデュケーション　2001 年

第１章　簡単なC++プログラム

課題１－１
次のように画面に文字を表示せよ。　以下、縦棒が左にある行は実行結果を示す。

私の名前は aaaa です。
bbbb 高校を卒業しました。
8 進数で o ???才です。　　（注）ここで　???は手計算。　???の前に o（オー）をつける
16 進数で x ???才です。　　（注）　同上　　???の前に x（エックス）をつける

課題１－２
次のように画面に文字を表示せよ。データはキーボードから入力し、それを表示する。

数値を 2 個入力してください。
データ A＝
データ B＝
値の大きい方は　　です。

課題１－３
次のように画面に表示せよ。データはキーボードから入力する。
ヒント：標準体重＝（身長－100）×0.9　　肥満度＝（体重－標準体重）÷標準体重

体重を入力してください。
身長を入力してください。
あなたの標準体重はｘｘｘｘｘです。
あなたの肥満度はｙｙｙｙｙｙです。

課題１－４
10 個の整数の最大値と最小値を求めるプログラムを作成せよ（データは配列に初期値で設定)。

課題１－５
10 個の実数をキーボードから入力し、ソートして大きい順に値を画面に表示せよ。

課題１－６
年を渡され、普通の年なら 0 を返し、うるう年なら 1 を返す関数　int is_leapyear (int y)　を作成せよ。

＊課題１－７
学生番号下 2 桁＋10～学生番号下 2 桁＋110 までの素数を出力するプログラムを作成せよ。
ヒント：素数は 1 と自分自身しか約数を持たない数。割り算をして余りがない数が自分だけなら素数。

第２章　関数　　ポインタ　　配列

課題　２－１

２個の変数の値を交換する関数　void swap2 (int* pa, int* pb)を作成せよ。

main 関数で値を入力し、swap2 関数を呼び出し、結果は main で表示する。

（注）swap という名前の関数はシステムに存在するので、swap2 という名前にする。

呼び出しは　int a,b;　とすると、　swap2(&a, &b);　となる。

整数を入力せよ。

a>>

b>>

交換すると

a=

b=

課題　２－２

２個の変数の値を交換する関数　void swap2 (int& x, int& y)を、参照を使って作成せよ。

main 関数で値を入力し、swap2 関数を呼び出し、結果は main で表示する。

呼び出しは　int a,b;　とすると、　swap2(a, b);　となる。

課題　２－３

3 つの int 型整数を昇順に並べかえる関数　　void　sort3(int* n1, int* n2, int* n3) { ……}

を作成せよ。

条件：sort3 関数は値の交換に課題２-1 の 2 個の値を交換する swap2 関数を利用する。

ヒント：　main 関数で、値をキーボードから入力し、sort3 関数を呼び出し、結果を表示する。

課題　２－４

関数　void keisan (int n1, int n2, int* wa, int* sa, int* seki)を作成せよ。

この関数は２個の整数 n1 と n2 の和、差、積を wa、sa、seki が指す変数に格納する。

2 個の整数を入力

整数 m>>

整数 n>>

m と n の和は

m と n の差は

m と n の積は

ヒント：main()関数で 2 個の整数を cin で読み込み、keisan 関数に渡し、結果を返してもらい、main 側で表示する。呼び出し　keisan(m, n, &a, &b, &c) ;　10 と 30 の差は-20 ではなく 20 である。

課題 ２－５
課題 ２－４を引数受け渡しのひとつの方法である参照で作成せよ。

課題　２－６　　配列とポインタの問題
5 個の整数を入力し、その要素の最大値と最小値を出力するプログラムを作成せよ。
ただし、5 個の整数を入力する関数（関数名：input）と最大値と最小値を求める関数（関数名：search）を作り、main 関数ではその 2 個の関数を呼び出し、結果を印字するようにせよ。
ヒント１：関数のインタフェースは次のようにする。
　　void input(int* pdata, int n)　 n はデータの個数で、ここでは 5。
　　void search(int* pdata, int n, int* pmax, int* pmin)
　ヒント２：main 関数で配列 data[N]を確保し、ここにデータを cin で 1 個ずつ読み込む。

＊課題　２－７
西暦年月日を入力し、その前の日を求めるプログラムを作成せよ。閏年を考慮すること。
関数としては、前日を求める関数（day_before）とうるう年を判定する関数（is_leapyear）を作成する。　　実行例：　2014 年 5 月 10 日　　⇒　5 月 9 日
ヒント１：各関数のインタフェースは次のようにする。
void day_before(int* py, int* pm, int* pd)　　//　引数として、年、月、日が渡されてくる。int is_leapyear(int y)　　// 年を渡され、普通の年なら 0 を返し、うるう年なら 1 を返す。
ヒント２：通常は年、月、日のうち、日から 1 を引けば前日であるが、
　　・日にちは 1 日が特別で、3 月 1 日の前日は 2 月 28(29)日になる。
　　・また月は 1 月が特別で、前の月は 12 月になる。
ヒント３：判定の方法としては月別に if 文で判定する

＊＊課題　２－８
上記の問題を月別に判定するのではなく、テーブルを持つ方法で作成せよ。
（テーブルを持つ方法をとるとエレガントなプログラムになる。）
ヒント：テーブルは下記のように月ごとの日数を持つ（うるう年を考慮し２パターン（２次元）になる）
int　day[2][13] = { { 0,31,28,31,30,....} , { 0,31,29,31,30,,,,} } ;
　　ここで、day[0][13]は普通の年、day[1][13]はうるう年の場合に使う。
　　たとえば、普通の年の 3 月 1 日の前の日は day[0][3-1]（つまり、day[0][2]=28）28 になる。

＊＊課題　２－９
西暦年月日を２個入力し、１個目と２個目の差が何日あるかを求めるプログラムを作成せよ。
これを用いて自分の誕生日から今日までの日にちを出せ。

第３章　文字と文字列

課題　３－１

要素数が 6 でデータ型が char 型である配列を文字列”ABCDE”で初期設定し、それを表示せよ。

文字列 str は ABCDE です。

課題　３－２

上記を書き換えて、代入文で 1 文字ずつ格納し、表示せよ。

　　ヒント：代入文　str[0]=‘A’;

文字列 str は ABCDE です。

課題　３－３

課題３－１の文字列を“ABC¥0DE”として初期設定して表示してみよ。¥は逆スラッシュ

文字列 str は ABC です。

課題　３－４

文字列（名前）を読み込んで、同じものを打ち出すプログラムを作成せよ。

　　ヒント：cin >> を使う。

名前を入力　ｘｘｘｘ

確認　ｘｘｘｘ　ですね。

課題　３－５

３個の文字列　“tandai” “tuusin” “shori” を 2 次元配列の char 型に初期値として設定し、各文字列を表示せよ。

　ヒント：　char str[ ][8]

str[0] = “tandai”
str[1] = “tuusin”
str[2] = “shori”

} ダブルクォーテーション「“」も表示すること。

課題　３－６　　前問で初期値として与えたものを標準入力（キーボード）から入力して、表示せよ。

>> tandai
>> tuusin
>> shori

} 入力データの表示

str[0] = “tandai”
str[1] = “tuusin”
str[2] = “shori”

} 「“」も表示する。

課題　3－7

関数　　void one__put_char (const char str[ ]) を作成せよ。

この関数は引数 str で渡された文字列を表示する。putchar 関数を用いて実現する。

ヒント：putchar 関数は 1 文字ずつ表示する関数なので、while 文で繰り返す必要がある。

　　終わりは'¥0'か 0 で判定する（while (str[i] !='¥0') と while(str[i]) は同じ意味になる)。

```
文字列を入力せよ：Tandai
Tandai
```

＊課題　3－8

関数　　void put_reverse (const char str[ ]) を作成せよ。

この関数は引数 str で渡された文字列を逆順に表示する。前問を応用する。

ヒント：main()関数で文字列を読み込み、put_reverse ()を呼ぶ。

　　put_reverse 関数はまず文字列の長さを見つける。長さを利用して逆から文字を表示する。

```
文字列を入力せよ：Tandai
逆にすると　iadnaT　である。
```

＊課題　3－9

関数　　int include_char (const char　str[ ]，　char c) {　　}　を作成せよ。

この関数は 1 番目の引数（文字列 str）の中に、2 番目の引数（文字 c）が含まれていれば、その何番目かを返す。同じ文字がある場合は最初に見つかった個所でよい。含まれていなければ、-1 を返す。

ヒント：main()関数を作り、include_char 関数を呼び出す。

　　main()関数では文字列と探す文字を cin で取り込む。次に include_char関数を呼び出す。

```
文字列を入力：ASDFGH
探す英大文字を入力：　S
それは 2 番目にありました。
文字列を入力：ASDFGH
探す英大文字を入力：　K
それはありませんでした。
```

＊＊課題　３－１０

名前の一覧から一致する名前を探し出す関数を作成する。この関数は一致する名前が名前一覧の何番目かを返す。一致する文字列がなかったら偽（0）を返す。

文字型配列で実現する。

条件

名前の一覧は２次元配列で定義する。チェックする名前はキーボードから入力する。

関数　　int nameNo(char p[ ][6], char s[ ]) { ....}

//　p[ ][ ]は文字型２次元配列（２次元目は大きさを指定する必要がある)、

//　s []はチェックする名前（文字列）

main()で名前一覧は定義する。

名前一覧、すなわち文字型配列で定義する名前（文字列）は下記とする。

なお、初期値の指定では内側の {} が無いと警告エラーになる。無くとも動作するが。

```
char name[ ][6] = {{ “Eluza”},{ “Candy”},{ “Emy”},{ “Ann”},{ “Rolla”},{ “Nancy”},{ ‘¥0’} };
nameNo(name,str)
```

```
チェックする文字列を入力してください。：Rolla
Rolla は 5 番目です。
チェックする文字列を入力してください：An
An はいません。
```

第４章　文字列とポインタ

課題　４－１

char 型の配列 st_ary に文字列”ARRAY”を初期値設定する。　　char str_ary[ ]= “ARRAY”;

char 型へのポインタ p が文字列“POINT”を指すように初期値設定する。　char *p=”POINT”;

それぞれが指している文字列を表示するプログラムを作成せよ。表示は cout を用いよ。

```
str_ary = “ARRAY”　　//　文字(")も出力する。
p = ”POINT”
```

課題　４－２

関数　　void one_put_char(const char *str ）を作成せよ。

この関数は引数 str で渡された文字列を表示する。ポインタ変数で実現せよ。

ヒント：　while (*str) {　cout << *str++ ;}

　　ここで　*str++は　*str の値がまず得られ、次にポインタがプラスされる。

```
文字列を入力せよ：Tandai
Tandai
```

課題　４－３

関数　　void　put_reverse(const char *str ）を作成せよ。

この関数は引数 str で渡された文字列を逆順に表示する。ポインタ変数で実現せよ。

ヒント：main 関数で文字列を読み込み、put_reverse 関数を呼ぶ。 put_reverse は最初に

　　while を使い、文字列の長さを見つける。長さを利用して逆から cout で表示する。

```
文字列を入力せよ：Tandai
逆にすると　iadnaT　である。
```

課題　４－４

関数　　int include_char (const char　*str,　char c) {　　}　を作成せよ。

この関数は 1 番目の引数　文字列 str の中に、2 番目の引数　文字 c が含まれていれば、その何番目かを返す。同じ文字がある場合は先頭でよい。含まれていなければ、-1 を返す。ただし、ポインタ変数で実現せよ。

ヒント：main()関数を作る。main()関数では初期値として文字列例えば、”ABCDEFGHIJKLMN”を持つ変数を用意する。また、探す英大文字を cin で char ch に入力させる。次に include_char 関数を呼び出す。返された値を判定して　何番目の文字か、または文字がなかった、のメッセージを出力する。

```
探す英大文字を入力してください：　M　　　←　例
それは 13 番目にありました。
```

課題 4 － 5

渡された文字列 str 内の全ての数字文字を削除する関数　void del_digit(char *str) {……}
をポインタ演算を用いて作成せよ。ただし、文字列は main()で　キーボードから入力する。

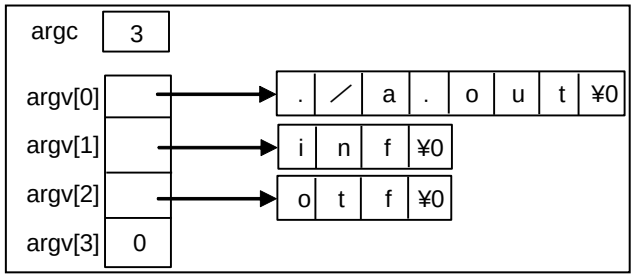
ABC123XYZ と入力すると、　ABCXYZ　と出力される。

課題 4 － 6

コマンド行の引数を出力するプログラム作成せよ。

（実行プログラム名：a.out の例）

```
$ g++  −o a.out  kadai.C
$ ./a.out  inf  otf
./a.out
inf                    結果の印字
otf
```



ヒント　　コマンド行とは？　　p５３５参照
$ ./kadai4-5　　//　引数なしのコマンド
$ ./kadai4-6　inf　otf　// 引数を持つコマンド
ここで、kadai4-6 の main 関数は　　int　main(int argc, char *argv[ ])　{.....}

課題 4 － 7

課題４－５　を、文字列をコマンド行から与えるように改造せよ。
ヒント：main()関数経由で文字列を del_digit へ渡す。

```
$./kadai4-7　 ABC123XYZ
ABCXYZ
```

＊課題 4 － 8

文字列 s1 と文字列 s2 が等しければ（先頭から'¥0'まで同じ）1（真）を、そうでなければ 0（偽）を返す関数　　int　str_equal(const char *s1, const char *s2) {　}　　を作成せよ。
main()関数は結果をもらって、結果を表示する。

＊課題 4 － 9

文字を整数型数値へ変換する関数　　int　strtoi (const char *str)　を作成せよ。
すなわち、数字の文字列を引数で渡すと、数値が返ってくる関数で、マイナスも考慮すること。
数字の文字列はキーボードから入力する。

＊＊課題　4 - 1 0
キーワードをバイナリ探索法で探索するプログラムを作成せよ。
キーワードは下記とする。
vI　aI　read　print　println　if　else　for　while　do　I　return　break

バイナリ探索法を使うためにはキーワードをテーブルに初期値として持つときに、アルファベット順にしておく必要がある。
　　char *keyTable[ ]={ “aI”,.....}
バイナリ探索法：先頭要素と最後尾要素との中間の要素を探索したいキーワードと比較する。
　　中間要素より小さければ、次に先頭要素と中間要素の間の探索に移る。
　　(中間要素を最後尾要素にして、同じことを繰り返す)。

＊＊課題 4 － 1 1
名前の一覧から一致する名前を探し出す関数を作成する。この関数は一致する名前が名前一覧の何番目かを返す。一致する文字列がなかったら偽（0）を返す。
名前の一覧は複数の文字列をポインタ配列で定義する。チェックする名前はキーボードから入力する。
　関数　　int nameNo(char *p[ ], char *s) { ....}
　　//　*p[ ]はポインタ配列で、 *s がチェックする名前（文字列）
main()で名前一覧は定義する。
名前一覧、すなわちポインタ配列で定義する名前（文字列）は下記とする。
　　char *name[ ] = { “Eluza”, “Candy”, “Emy”, “Ann”, “Rolla”, “Nancy”, NULL };

チェックする文字列を入力してください。：Rolla
Rolla は 5 番目です。
チェックする文字列を入力してください：An
An はいません。

第５章　構造体

課題　５－１

int 型、long 型、double 型の３つのメンバを持つ構造体（名前：Kouzou）を struct で定義し、各メンバに値を代入して表示せよ。変数名は data とする。(注) 変数名と構造体名を混同しないように。

main()関数内に代入と表示を記述する。

実行結果
data.i = 10
data.l = 40000
data.d = 1.4142

課題　５－２

前問の構造体を持つ変数(data)と、それを指すポインタ(p)を以下のように定義する。

　　Kouzou　data = { ..... } ; 構造体の変数に初期値を設定する

　　Kouzou*　p =　&data ; ポインタの定義、初期値として data のアドレスを持つ

ポインタが指す変数の各メンバの値を表示するプログラムを作成せよ。　p->i　は　(*p).i　に同じ。

実行結果
p->i = 10
p->l = 40000
p->d = 1.4142

課題　５－３

関数　　void set (Kouzou* pp, int a, long b, double c ）を作成せよ。

この関数はポインタ変数 pp の構造体（Kouzou 型）の各メンバに、引数で渡された a,b,c の値を格納する。

なお、構造体の定義は set 関数、main 関数の前に記述する（こうすると関数間共通になる)。

ヒント：main 関数での呼び出しは set(&data, 10, 40000, 1,4142 ）　で、set 関数内では pp->i=a で値をセットする。結果の表示は main 関数で行う。

実行結果
data.i = 10
data.l = 40000
data.d = 1.4142

課題　５－４

構造体を typedef で表現して、前問のプログラムを作成せよ。typedef で用いる名前は KOUZOU とする。　なお、typedef の定義は関数 set、main( )の前で行う。(KOUZOU は小文字の kouzou でもよいが、typedef では大文字を使うことが多い)。

ヒント：　typedef を用いたデータ型の定義は次のように行う。

```
typedef  struct {     // このように定義すると     KOUZOU  data;  と書ける。
                int i;  long l;  double d;
            } KOUZOU;
```

```
data.i = 10
data.l = 40000
data.d = 1.4142
```

課題　５－５

関数　　KOUZOU　set (int a, long b, double c) を作成せよ。　　typedef を使うこと。

この関数は KOUZOU 型の変数 temp を持ち、各メンバに引数で渡された a,b,c の値を格納し、関数の戻り値で値を返す。　ヒント：main 関数での set 関数呼び出しは data = set(10, 40000, 1,4142);という代入文になる。　set 関数内では temp.i=a　でセットする。

```
data.i = 10
data.l = 40000
data.d = 1.4142
```

課題　５－６　　構造体の配列

５人分の名前と点数を入力し、点数の高い順に並べ替えて画面に表示せよ。

人数は後で変更になることを考慮し、#define N 5 で定義する。

```
構造体　　typedef  struct {
                      char name[10],
                      int  tensuu;
                   }  SEISEKI ;
```

typedef 宣言は swap2()関数、sort()関数、main()関数の外側で行う（全部の関数で使える)。

main()関数の仕様：データを入力。sort()関数を呼び出す。結果を出力する。

sort()関数の仕様：void sort(SEISEKI data[ ], int n)　　n は人数。

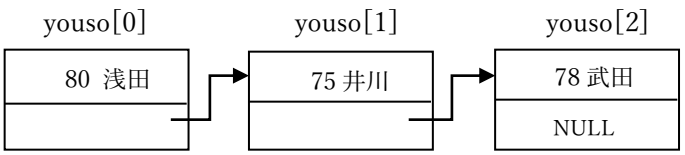
点数を判定して並べ替える。並べ替えは swap2()関数を作成し、呼び出す。

swap2()関数の仕様：void swap2(SEISEKI * pa, SEISEKI * pb)

構造体の要素を交換する関数。

ヒント：構造体の要素は個々ではなく一度に要素単位で可能。

課題　５－７　連結リスト（データの途中挿入などに便利なデータ構造）
次のプログラムは連結リストの処理である。
このプログラムは 5 個の要素を持つ変数 youso を定義し、入力データを youso[0]から順に読み込み、出力も youso[0]から順に出している。このプログラムを修正して、入力データとして名前と点数が入るようにし（int data; を SEISEKI　data;　とする)、点数がマイナスならば終了。
テスト：では 3 人分を入れて、それを表示せよ。




```
#include <iostream>
using namespace std;
int main(){
        int i;
        struct List{      // リストのデータ構造
                int data;
                List *ptr;   // 自己参照型ポインタ
        };
        List   youso[5];      // 構造体の配列
        List   *head,         // 「リストの先頭へのポインタ」
                    *curr,          // 「現在要素へのポインタ」
               *next ;         // 「次の要素へのポインタ」
        curr=head=next=youso;   // yousoの先頭をセットする。
// リスト作成を作成する
        for(i=0; i<5; i++){      // 要素5個分ループ
                cin >> next->data;   //データの読込み
                if (next->data<0)
                        break;   // マイナスが来たら終了
                curr=next;   // データを入力した要素を「現在要素へのポインタ」へセット
                [          ]  // 「次要素へポインタ」を更新。それを「現在要素へのポインタにセット
        }
        curr->ptr=NULL;      //   最後の要素にＮＵＬＬをセットする
// リストを表示する
        curr=head;
        cout << "出力します¥n";
        do{       cout << "data= " << curr->data << endl;
                curr=curr->ptr;
        } while(curr!=NULL);
        return EXIT_SUCCESS;     // return 0;
}
```

実行結果
点数>>80
名前>>浅田
点数>>75
名前>>井川
（３人分入力）
点数>>－１
出力します
浅田　　80
井川　　75
（３人分出力）

課題　５－８　連結リスト（逆順リスト）

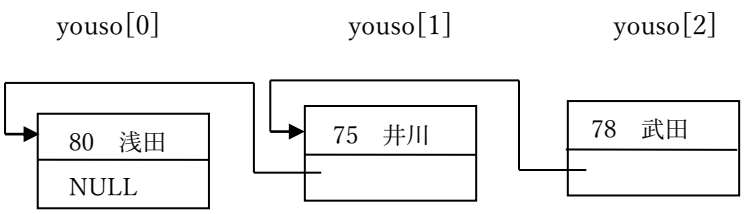
課題５－７のプログラムはリストが順にリンクしているが､これを逆順にリンクするように修正せよ。テストはデータを入力して、それを逆順に出力する。

ポインタが逆順になっているので出力はポインタに沿って行えば自然に逆に表示される。

ヒント：リストは youso[0][1][2]の順に作成される。youso[1]を作るときに youso[0]を指すようにする。

tail ポインタを用意し、常に最後尾のリストを指すようにする。

出力時は tail ポインタをスタートにして逆順にたどる。



実行結果

```
点数>>80
名前>>浅田
点数>>75
名前>>井川
（ｎ人分入力）
点数>>－１
出力します
（ｎ人分出力）
井川　　75
浅田　　80
```

ヒント：ptr には前の要素へのポイントを入れる。

＊＊課題　５－９

課題５－７を元に順リスト構造を操作する append 関数を作成する。

①データをリスト構造の最後尾に追加する関数 append()を作成せよ。

仕様：　List* append (List* wp, List* tp)

　　wp は追加するデータを指すポインタ、tp はリストの最後尾を指す tail ポインタ。

　　返す値は更新された最後尾へのポインタ。

なお、リスト構造の変数は main()の中に配列で確保する。

この関数を用いて、点数と名前（英字名）を登録し、表示せよ。点数がマイナスなら入力を終了する。なお、最初の１個は main()関数で設定すること。２個目から append()関数を呼ぶ。

＊＊課題　５－１０

課題５－９に次の関数を追加せよ。

②データをリストの途中に挿入する関数 insert()を作成する。

仕様：　List*　insert (List* wp, List* hp)

　　wp は追加するデータを指すポインタ、hp はリストの先頭を指す head ポインタ。

テストはまずアルファベット順にデータを登録する。

次に最後尾にデータを追加する。　　append 関数のテスト。

そして、途中にデータを追加する。　　insert 関数のテスト。

最後にリストの先頭にデータを挿入する。insert 関数のテスト。

データを表示する。

ヒント：main()関数では名前(英字名)を比較して append 関数を呼ぶか insert 関数を呼ぶかを判定する必要がある。アルファベットが最後尾より大きければ append 関数を呼ぶ。

第６章　動的なメモリ

課題　６－１

次のリスト構造のデータ領域を動的に確保し、データを格納せよ（データは整数値と NULL ポインタ)。次に格納したデータを表示し、最後にメモリを delete 演算子で解放せよ。

```
struct　List {　　// リストのデータ構造
    int　data;
    List　*ptr;　　// 自己参照型ポインタ
}
```

課題　６－２

課題６－１のリスト構造を２個、動的に作成し、リンクさせよ。

最初の１個を動的に確保したら、整数値と NULL を設定し、次の２個目を動的に確保したら、整数値と NULL を設定し、１個目のポインタ部には２個目のアドレス（new 演算子の返す値）を設定する。

１個目のデータ(整数値)と２個目のデータ(整数値)を表示せよ。ただし、１個目から２個目へはリンクをたどっていき、表示する。最後にメモリを２個とも delete 演算子で解放する。

＊課題　６－３　　動的な連結リスト

上記を N 個について行うこと（入力個数が不定)。入力データは名前と点数で、点数がマイナスの値で終了する。

```
struct　List {　　　　// リストのデータ構造
    char　name[10];　//　名前
    int　　tennsuu;　　//　点数
    List　*ptr;　　　　// 自己参照型ポインタ
}
```

入力データは入力順にリストに追加されていく。１個入力されるごとに１個のリスト型のデータを new 演算子で確保する。また、メモリの解放（delete）は new 演算子で作成したすべてのメモリについて行う。

＊＊課題　６－４

動的な連結リストに関する append()関数、insert()関数を作成せよ。インタフェースは課題５－９、５－１０を参照。リストのデータは名前がアルファベット順になるようにする。

ヒント：main 関数で append()関数、insert()関数を使いアルファベット順に登録していく。５－１０参照。

第７章　プロセスの生成（システムコール　fork,wait,exit,exec ）

課題 ７－１　以下のプログラムをコンパイル､実行すると どういうことが起こるか｡図解せよ。

```
#include <iostream>
using namespace std;
int main( ) { int I,j,k;
        cout << "fork1¥n";
        i=fork();
        cout << "fork2 i=" << i << "¥n";
        j=fork();
        cout << "fork3 j=" << j << "¥n";
        k=fork();
        cout << "last k=" << k << "¥n";
        return 0;
}
```

課題　７－２ 次の２個のプログラム（A)、(B）を作成・実行せよ。 違う個所をチェックし、違う理由を述べよ。

（A)

```
#include <iostream>
#include <sys/wait.h>      // wait を使うときはこのヘッダーをインクルードする。
using namespace std;
int main(int argc,char *argv[ ]) {
        int i,dep_time,pn,st;
        i=atoi(argv[1]);
        dep_time=i*60;
        if((pn=fork())==0){
                cout << "子プロセス　始まる。 pn=" << pn << endl;
                sleep(dep_time);
                cout << i << " 分経過しました。子プロセス終了。" << endl;
                exit (0);
        }else {    cout << "親プロセス　始まる。 pn=" << pn << endl;
                wait(&st);}
        cout << "プロセス終了７－２－２。" << endl;
        return 0;
}
```

（B）

```
#include <iostream>
using namespace std;
int main(int argc,char *argv[ ]) {
        int i,dep_time,pn;
        i=atoi(argv[1]);
        dep_time=i*60;
        if((pn=fork())==0){
                cout << "子プロセス　始まる。 pn=" << pn << endl;
                sleep(dep_time);
                cout << i << " 分経過しました。子プロセス終了。" << endl;
        }else { cout << "親プロセス　始まる。 pn=" << pn << endl; }
        cout << "プロセス終了７－２－１。" << endl;
        return 0;
}
```

＊課題　７－３

前の課題を参考にして、fork でプロセスを作り，親プロセスと子プロセスが交互に実行（１文字出力）を繰り返すプログラムを作成せよ。

ヒント：子プロセスに実行が移った時に 1 文字出力。親プロセスは wait していて、プロセスが戻ったら 1 文字出力する。また，合計何文字出力するかはコマンド行の引数で指定できるようにする。このとき、何文字まで指定できるようにするか仕様を明確にして、文字数をチェックすること。また、奇数が指定された時はエラーを表示する。　なお、012...　　ABC..　　などの文字は初期値で持つ。

```
$　./kadai17-3　10
0A1B2C3D4E
```

ヒント：実行してみて、結果が思ったように出なかったら、cout << flush;を入れてみる。
cout はバッファに値が残るので flush で強制出力する必要があるため。

課題 7－4　次のプログラム（簡単なシェル)を exec を使って作成し、実行せよ。

$ ./kadai7-4
> /usr/bin/cal　　//簡易シェルの中の execl は　execl(“/usr/bin/cal”, /usr/bin/cal”, NULL);
になる。

```
#include <iostream>
#include<sys/types.h>
#include<wait.h>
#include<unistd.h>
using namespace std;
int main(){　　//　簡易シェルプログラム
　　char　command[20];
　　int st;
　　static char prompt[4]="> ";
　　cout << prompt;
　　while(!cin.eof()){　　//　cin.eof()は Ctrl+D が入力されたかを判定する。
　　　　cin >>　command;　　//　cin は最初の単語しか読み込まない。
　　　　if(　?　　){　// 子プロセス
　　　　　　if(execl(command,command,NULL)<0){　//　失敗ならマイナス
　　　　　　　　?　　}　　//　失敗　exit(1)
　　　　}else{　//親プロセス
　　　　　　?　}
　　　　cout << prompt;
　　}
　　cout << endl;
　　return EXIT_SUCCESS;
}
```

＊課題 7－5 次のプログラムは課題 7-4 を拡張している。
このプログラムのテストデータを作成して、実行せよ。
getarg 関数の出力データを図解して提出せよ（引数の arg と buf の関係を図示する)。

$ ./kadai7-5
> /bin/ls　-l　testfile　　// testfile はあらかじめ作成しておく必要がある。
> /usr/bin/cal　6 2017　　// コマンドに引数があることに注意する。

```
//    getarg 関数は空白で区切られた文字列をポインタの配列に分解する
int getarg(char *argv[],char *buf){
        int i;
        for(i=0; *buf !='¥0'; i++){        //buf が'¥0'になるまで
                while(*buf==' '){              //空白
                        *buf='¥0';
                        buf++;
                }
                if (*buf=='¥0'){   // buf が'¥0'なら終わる
                        break;
                }
                argv[i]=buf;      //arg[i]から buf の文字列を指す。
                while( (*buf != '¥0') && (*buf   != ' ') ){
                        buf++;
                }
        }
        argv[i]=NULL;
        return i;
}
```

```
int main(){              //少し複雑なことを処理できる簡易シェル
        char *argv[256];
        char line[256];
        int st;
        static char prompt[4]="> ";
        cout << prompt;
        cin.getline(line,256);   // cin.getline は 1 行分の文字列を引数 line に読み込む。
                                 // 256 は制限値。
                                 // 何も入力されなかったら、ヌル文字が入ってくる。
        while( *line !='¥0'){
                getarg(argv,line);     //上記の関数。
                if(                    ){
                        if(execv(argv[0],argv)<0){      //   execv が失敗ならマイナス
                                        ?
                        }
                }else{
                                ?
                }
                cout << prompt;
                cin.getline(line,256);
        }
        return EXIT_SUCCESS;
}
```

課題 7－6　下記のプログラムを改造して次ページのようなメモリマップを出力せよ。
外部変数 environ には環境変数が格納されている文字列へのポインタが格納されている。

```
#include <iostream>
using   namespace std;
extern char **environ;
int      data0,data1=10;
void   func( ){
            int i;
            cout << &i << "スタック func 内の変数" <<   endl;
}
int   main(int argc,char *argv[]) {
            int i;
            cout << "--------------¥t-------" << endl;
                  printf (“%p¥t main 関数のアドレス”, main)
            cout <<            <<"¥t|初期値あり" <<   endl;
            cout <<            <<"¥t|初期値なし" << endl;
            cout << new int <<"ヒープ領域" << endl;
            func( );
            cout <<        <<"スタック main 内の変数" <<   endl;
            cout << argc        <<"argc" << endl;
            cout << environ << "¥t 環境変数" << endl;
            return 0;
}
```

メモリマップ（例）

```
[katogi@host00 cpp]$ ./nkadai7-6 10 20 30                    ←　コマンドに引数を 3 個つけた場合
-----------------------------------------
アドレス          |      内       容
-----------------------------------------
0x804864c      | 共有ライブラリ  printf
-----------------------------------------
0x80487e6      | func 関数
-----------------------------------------
0x8048822      | main 関数
-----------------------------------------
0x804a194      | 初期値あり　変数 data1     4byte              ←　4 も出力する
-----------------------------------------
0x804a230      | 初期値なし　変数 data0
-----------------------------------------
0x9f7c008      | ヒープ領域　int 型          4byte
-----------------------------------------
                |
-----------------------------------------
0xbfa92134     | スタック func 関数の変数 i
-----------------------------------------
0xbfa92158     | スタック main 関数の変数 i
-----------------------------------------
0xbfa92180     | argc          argc の値(4)                   ←　argc の値は　4　になる。
0xbfa92204     | argv
-----------------------------------------
0xbfa92218     | 環境変数
-----------------------------------------
```

第８章　ファイル処理

課題　８－１

キーボードから入力ファイル名を入力し、そのファイルが存在すれば「ファイルは存在します」と表示し、そうでなければ「ファイルは存在しません」と表示するプログラムを作成せよ。

ヒント： fstream を使用する。

```
ファイル名は？　ｘｘｘ
ファイルは存在します。
ファイル名は？　yyy
ファイルは存在しません。
```

ファイル keisoku

Yoshida	160	59
Suzuki	180	76
Asada	176	80
Ito	172	65
Tasaki	165	58

課題　８－２　キーボード　から　ファイル　へ

キーボードから、名前、身長、体重を入力し、それをファイルに書き込むプログラムを作成せよ。

ファイル名は keisoku とする。書き込むファイル名はキーボードから入力する。

ヒント：1 人分を入力し、続けるか否かを問い合わせ、続けるなら次の人を入力する。続けるか否かは、1 が入力されたら続ける。0 が入力されたら終わる。

課題　８－３

キーボードからファイル名（keisoku）を入力し、そのファイル中の行数(改行文字の個数)をカウントして画面に表示するプログラムを作成せよ。

課題　８－４

コピー元ファイル名（keisoku)、コピー先ファイル名（任意）をキーボードから入力し、ファイル keisoku の英小文字を英大文字に変換し、コピー先ファイルに書き込むプログラムを作成せよ。結果は more コマンドで表示せよ。

課題　８－５

ファイル keisoku から名前、身長、体重を読み込み、身長でソートして結果を画面に表示せよ。また、平均身長と平均体重を表示せよ。

main 関数にすべてを記述するのではなく、関数に分けて作成せよ。

作成すべき関数は 2 個のデータを交換する swap 関数、データを並べ替える sort 関数とする。

ヒント：読み取ったデータは構造体にセットするとよい。

```
typedef　struct {
        char　name[100];　　double height;　　double weight;
    } HITO_TYPE ;
```

＊課題　8－6　トークン作成

（1）ファイル作成

右の内容をキーボードから行単位で入力し、ファイルに出力するプログラムを作成せよ。

ファイルの名前は prog.C とする。

```
int main( ) {
   if ( abc124 > 100 )
      flag = 0 ;
   return ;
}
```

（2）prog.C ファイルを読込み、中身を「トークン」に分けて、各々を配列テーブルに格納し、それらを表示せよ。

「トークン」とは区切り記号（空白、改行）で区切られた識別子（変数名、キーワード、関数名など)、数値、記号（演算子など）である。

○識別子は最初の 1 文字が英字で、区切り記号が来るまで、英字又は数字が続く。

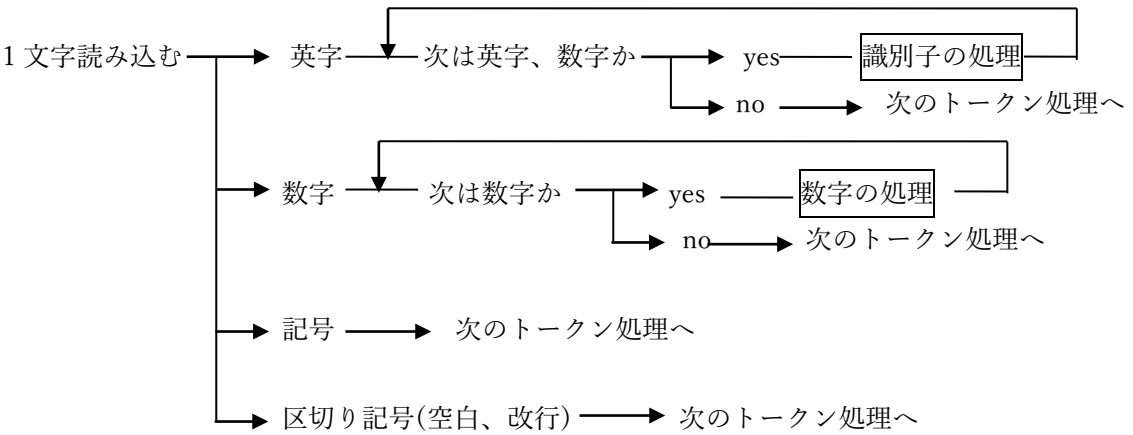
　int　main　if　abc 124　flg　return

○数字は先頭が数字で、区切り記号が来るまで、数字が続く。

　100　0

○記号は 1 文字である。　　（　）｛　｝　＞　＝　；

＜トークン処理＞　アルゴリズムは最初の 1 文字で判断・分岐し、各処理を行う。




```
$ ./kadai8-6-2　prog.C
識別子
　int　main　if　abc 124　flg　return
数字
　100　0
記号
　　（　）｛　｝　＞　＝　；
```

第９章　オブジェクト指向

９．１　クラス

課題９-1　　People クラスと main 関数を作成して実行させよ。

People クラス　　//　住所、名前を格納するクラス。
　　データメンバ　　//　private：他のクラスからは見えなくする。
　　名前　　// name　　文字配列型
　　住所　　// address　　文字配列型
　　メンバ関数　　// public：他のクラスから見える。つまり、呼び出すことができる。
　　名前を登録する。　　//void　setName(引数：文字配列型　)　インライン関数
　　住所を登録する。　　// void　setAddress(　引数：文字配列型　)　同上
　　名前と住所を表示する。　// void　show( )

main 関数の手順
　　People クラスのオブジェクトを生成する。
　　住所と名前をキーボードから入力する。
　　People クラスのオブジェクトに対して setName 関数などで名前、住所を登録する。
　　People クラスのオブジェクトを表示する。

９．２　　配列、コンストラクタ

課題９-２　次の成績表クラスと main 関数を作成せよ。
（１）入力するデータは１人分で、コンストラクタの引数で与えること。
　　＝＝プログラム提出必要なし

成績表クラス
　　データメンバ　// private
　　氏名　　//文字配列型
　　英語の点数　　//整数型
　　数学の点数　　//整数型
　　メンバ関数　　/ /public
　　コンストラクタ　　//引数が３個あるコンストラクタ
　　//氏名、英語の点数、数学の点数をセットする。
　　表示関数　　//氏名、英語の点数、数学の点数を表示する。

main 関数
　　成績表クラスから１個のオブジェクトを作り、コンストラクタで初期値を与え、表示する。

（2） 入力するデータは 3 人分で、コンストラクタの引数で与えること。
＝＝プログラムと結果を提出せよ

main 関数
成績表クラスから 3 個のオブジェクトの配列を作り、コンストラクタで初期値を与える。
(例) Seiseki sk[3]={ Seiseki(“近藤”,80,68), Seiseki("土方”,78,90), ・・・ }
for 文を使って sk[i] を表示する。

9．3 継承関係（汎化／特化）

課題９－３ 継承関係のある 3 個のクラスを作成せよ。＝＝プログラム、結果の提出必要なし

People クラス 課題９-1 をそのまま使う

Student クラス （People クラスの派生クラス）
データメンバ // private
学生番号 // id 整数型
所属学科 // course 文字配列型
学年 // year 整数型
メンバ関数 // public
学生番号を登録する
所属学科を登録する
学年を登録する
名前、住所、学生番号、所属学科、学年を表示する。 // show
(ただし、名前と住所は親の People クラスのメンバ関数 show を呼び出す)

Teacher クラス （People クラスの派生クラス）
データメンバ 所属学科 // course
授業科目 // subject
メンバ関数 所属学科を登録する
授業科目を登録する
名前、住所、所属学科、授業科目を表示する。 // show
(ただし、名前と住所は親の People クラスのメンバ関数 show を呼び出す)

main 関数
Student クラスからオブジェクトを生成する。
Student のオブジェクトに対して名前、住所、学生番号、所属学科、学年を各々登録する。
Teacher クラスからオブジェクトを生成する。
Teacher のオブジェクトに対して名前、住所、所属学科、授業科目を各々登録する。
学生の住所、名前、学生番号、所属学科、学年を表示する。
先生の名前、住所、所属学科、授業科目を表示する。

課題９-４　課題９-３をもとにして、各々N 人を登録できるように改造し、結果を表示せよ。
　　なお、データはキーボードから入力する。＝＝プログラムと結果を提出せよ。
ヒント：配列変数を用いる。配列の大きさは N とする(#define を使う)。テストは 2 名でよい。

＊課題９-５（スポーツクラブの）会員クラスと会社クラスとを作成せよ。
　　スポーツクラブの会員はどこかの会社に勤務しているものとする。
main 関数は会社データ 2 社（a 社、b 社）とスポーツ会員データ 3 人分（x 氏、y,z 氏）のデータをキーボードから入力し、その後入力した会社データと会員データを表示する。
ただし、データとして　x 氏は a 社、y,z 氏は b 社に所属する。
（この課題は継承関係はないが、関連するクラスを作成するものである)。

スポーツ会員クラス		
データメンバ	勤務先名（＝会社名）　氏名　住所	// private
メンバ関数	データメンバにデータをセットする関数	// public
	勤務先名をゲットする関数	//　char*　getKinmu(){ return 勤務先名}
	会員名をゲットする関数	// char*　getName(){ return 会員名 }
	データメンバを表示する関数	//　void show( ) { ..... }

会社クラス		
データメンバ	会社名　資本金　売上	// private
メンバ関数	データメンバにデータをセットする関数	//データは一括セット
	会社名をゲットする関数	// char*　getShamai(){ return 会社名 }
	データメンバを表示する関数	//void show()

＊課題９-６　課題９-５の main 関数を改造し、特定の会員名を入力し、その会員に関するすべてのデータを表示せよ。

例えば、x 氏と入力すると、
x 氏の会員データとして「氏名、住所、勤務先」、勤務する会社のデータ「会社名、資本金、売り上げ」を表示する。

条件
結果の表示は会員クラスの show()を改造する。show の引数に会社クラスを設定する。
　　会員クラス　　void show(会社クラス　kaisha) { ..... }
会員クラスの show()は、引数として会社クラスを受け取り、show()の中では引数のデータを用いて会社クラスの show()にアクセスする。

１０　オブジェクト指向・総合課題

＊課題１０－１　　おつりのお金の種類と枚数を求めるプログラムを作成せよ。
①買い物は 1000 円以下とする。(支払いは 1000 円以下)。5000 円、10000 円での支払いはない。
②おつりは 500 円硬貨、100 円硬貨、50 円硬貨、10 円硬貨、5 円硬貨、1 円硬貨とする。
例えば、550 円の品物を購入すると、1000 円を支払えば、おつりは 450 円で、100 円硬貨 4 枚、50 円硬貨 1 枚となる。

Money クラス
　データメンバ　//private
　　char name[20];　// 500 円硬貨等、硬貨の名称をセットする
　　int　unit;　　// 500 等、硬貨の単位を入れる。
　　int　num;　　// 各硬貨のおつり枚数をセットする。
　メンバ関数　　//public
　　コンストラクタ　　// 引数 2 個（硬貨の名称、単位）を受け取り
　　　　// 硬貨の名称、硬貨単位、枚数＝０　の初期化を行う。
　　名称、単位、枚数を get する get 関数を定義する。
　　枚数をセットする set 関数を定義する。

Oturi クラス
　データメンバ　　なし
　メンバ関数　（静的メンバ関数とする）
　　keisan 関数（支払い額、購入額、マネー情報）//支払額－購入額　からおつりを計算.

main()関数
　Money クラスのインスタンス作成(配列)各インスタンスに硬貨の名称、単位をセットする
　支払額と購入額のデータをキーボードから入力する
　keisan（支払い額、購入額、Money クラスのインスタンス）を呼び出す。
　　//　Money クラスのインスタンスにおつり枚数の結果がセットされて、戻ってくる
　表示する

実行結果

```
購入金額は？　　250
支払い金額は？　1000
500 円硬貨　　1 枚
100 円硬貨　　2 枚
50 円硬貨　　1 枚
```

課題１０-２　　次のクラスを作成せよ。　　オブジェクト配列を使う　（学生と先生のクラスと同じ）

車クラス　　// class　Car
　　データメンバ　//private
　　　　車番　　車種
　　メンバ関数　　//public
　　　　車番と車種を登録する。
　　　　車番と車種を表示する。

パトカークラス（車クラスの派生クラス）　　// class Patcar :public Car
　　データメンバ　　//private
　　　　配属署
　　メンバ関数　　//public
　　　　配属署を登録する
　　　　パトカーの情報を表示する（車番、車種、所属署)。

自家用車クラス（車クラスの派生クラス）　　// class Mycar :public Car
　　データメンバ　//private
　　　　所有者名
　　メンバ関数　　//public
　　　　所有者名を登録する
　　　　自家用車の情報を表示する（車番、車種、所有者名)。

main 関数
　　　　複数のパトカーのデータ（車種、車番、配属署）と
　　　　　自家用車のデータ（車番、車種、所有者）をキーボードから N 台入力する。
　　　　入力したデータを登録する。(クラスから作成したオブジェクトに登録する)。
　　　　パトカーと自家用車のデータを表示する。

実行：テストデータは２台ずつでよい。

注意：データ入力後にはまだバッファには最後のデータ（例えば：Ctr+D）が残っている。
　　バッファをクリアしておかなければ、次の入力に影響するため、cin.clear()でバッファをクリアしなければならない。

課題１０-３　不定個数データの入力

課題１０-２を改造して、パトカーN 台を登録した場合、登録された台数を求めるプログラムを作成せよ。データはキーボードから入力し、Ctr＋D で入力を終了する。
表示は台数と登録されたパトカーの情報とする。
ただし、台数は、静的メンバとして記憶する。
＊自家用車に関しては改造の必要なし。

車クラスの改造

```
車クラス
    データメンバ
        車番        //  private
        車種        //  private
        static     パトカー台数    // public     static は  public  にする必要がある。
    メンバ関数        // public
        車番と車種を登録する。      //set 関数
        車番と車種を表示する。      //show 関数
      static  パトカーの台数を表示する     //  static void  showPatDaisu {  台数を表示する }
```

main 関数の改造（キーボードからのパトカー・データ入力の部分）

```
    ・・・・
  車番を入力する（キーボードから）
  while(入力終了か)  {       //  while( !cin.eof() )   Ctr+D が来るまで繰り返す。
        車種を入力する。
        車番と車種を登録する。
        パトカーの所属署を入力、登録する。
        パトカー台数をカウントアップする。      //    Car::patDaisu++；
        車番を入力する。
    }
    台数を表示する。
    パトカーの情報を表示する。
```

＊課題１０-４　データ検索

課題１０-３を改造して、パトカーの車番を与えたらどこの署に配属されているかを求めるプログラムを作成せよ。
ヒント：パトカークラスの中に「パトカーの配属署はどこ」というメンバ関数を作成。
　　引数は車番とする。

```
パトカークラス（ただし、自動車クラスの派生クラス）　{　 // class Patcar:public Car
    データメンバ　 //private
        配属署　　 //文字配列型
    メンバ関数　　 //public
        配属署を登録する
        void　パトカーの情報を表示する（ ）｛　　｝
        char*　パトカーの配属署はどこ（int 検索する車番）　 {
            検索する車番が一致したら、配属署を返す。
                                不一致なら、NULL を返す。　}
}
```

main 関数の改造

```
パトカーのオブジェクトを作成する。　// Patcar　pt[10];
パトカーのデータを登録する。　// pt[i].shabanSet(車番)；　　車番はキーボードからの入力
車番を与えてパトカーの配属署名を見つける。　// szk=pt[i].whatShozoku(検索する車番)；
なければ、なかった旨を表示する。
```

（例）　実行例

```
＝＝パトカーの配属署を検索する。＝＝
検索車番は >>　 3333
ひたちなか署です。
```

＊課題１０－５　　（キーボードから入力　＝＞　ファイルへ出力）

（1）パトカーN 台のデータをファイルに登録するプログラムを作成せよ。
　　ファイル名はキーボードから入れる。入力データの終了は Ctr+D とする。

パトカーファイル

```
車番　　車種　　配属署
車番　　車種　　配属署
・・・・・・・・・・・・・・
```

（2）（ファイル入力　＝＞　台数の画面表示）

課題１０-４　はデータをキーボードから入力しているが、これを上記の課題で作成したファイルからデータを読み込むように書き換えよ。

（3）検索の課題

上記の（2）に、警察署に配属されているパトカーのある車種の台数を求めるプログラムを追加せよ。

（実行例）

```
//　ファイルへパトカーのデータを書き込む。　　１０－５（１）のプログラムの実行
```

```
//　ファイルからパトカーのデータを読み込む。　１０－５（２）のプログラムの実行
パトカーの車種調査：
調査したい車種を入力せよ。　>>　クラウン
2 台
上記外に存在する車種は？
フェアレディ
BMW
```

課題１０－６　　（キーボードから入力　＝＞　ファイルへ出力）

キーボードから数人（不特定多数）の名前、身長、体重を入力し、ファイル sokutei に書き込むプログラムを作成せよ。キーボードからの入力終了は各自考える（例：0 入力、あるいは Ctrl+D キーなど)。

課題１０－７

上記で作成したファイルからデータを読み込み､各人の標準体重と肥満度を表示するプログラムを下記の仕様に沿って作成せよ。getName 関数等の仕様（引数のデータ型、個数、戻りのデータ型）をどうするかは各自で設計する。

```
class　Person //ある人の身長と体重を保持する。身長と体重から標準体重と肥満度を計算する。
    データメンバ　　名前、身長、体重
    メンバ関数　　// get 関数、set 関数はインライン関数にする
            getName( )　 getWeight( )　 getHeight( )
                //データメンバの名前、体重、身長を返す関数
            setName( )　 setWeight( )　 setHight( )
                // 引数で渡された名前、体重、身長をセットする関数
            calcStdWeight( )
                //　標準体重を計算する関数　　標準体重＝（身長-100）×0.9
            calcHimando( )
                //　肥満度を計算する関数　　肥満度＝（体重-標準体重）／標準体重
main 関数
    Person クラスのオブジェクトを作成する。　　// Person　p[100];
    ・ファイルから全員分のデータを Person のオブジェクトに取り込む。
    ・各人の標準体重(double 型)、肥満度(double 型)を計算する。
      ただし、この計算は Person クラスのメンバ関数を呼び出す（ことで実行する)。
    ・計算した結果を表示する。
```

```
//実行結果
読み込むファイル名は？
sokutei
○○さんの標準体重は・・・・で、肥満度は・・・・・です。
```

課題１０－８

上記の課題を修正して、計算結果をファイル(kekka)に出力するプログラムを作成せよ。
ファイルでの並びは「名前、身長、体重、標準体重、肥満度」とする。

総合課題１１　オブジェクト指向　機能設計とオブジェクト指向設計

課題１１－１　次の要求仕様（ビデオショップ）の C++プログラムを作成せよ。―機能設計―

要求仕様　　次のコマンドを処理するプログラムを作成。

$　rentalShop　　業務種別　　メディア種別　　タイトル　　会員名

業務種別：rent、back、(追加：sell)　　メディア種別：video、cd

使用例

$　rentalShop　rent　video　xxxxxx　aoki

//aoki ですが、ビデオを借りたいのですが。

aoki さん　タイトル xxxxx の video をお貸しします。返却日は 1 週間後です。

$　rentalShop　rent　video　zzzzz　aoki

aoki さん　タイトル zzzzz の video は貸し出し中です。

$　rentalShop　back　cd　yyyyyy　tanaka

tanaka さん　タイトル yyyyyy の cd 返却承りました。有難うございました。

データファイル

ファイル仕様は次のようにする。　このファイルはあらかじめ vi 等で作っておく。

タイトルは意味のあるものにする。

	メディア種別	タイトル	状態	貸出先		メディア種別	タイトル	状態	貸出先
初期	video	xxxxx	stock	none	実行後	video	xxxxx	rent	aoki
	cd	yyyyy	rent	tanaka		cd	yyyyy	stock	none
	video	zzzzz	rent	tanaka		video	zzzzz	rent	tanaka

ヒント：次のようなクラスを作る。但し、解答は上記のクラス分けにこだわらなくてもよい。

■class RentalApp

属性：メディア種別、タイトル、貸し出しまたは返却の種別、会員名を含むデータ型

操作：　借りるための関数：rent　　返すための関数：back　を作成する。

各関数は Util クラスの changeItemDBStatus を呼び出す。

■class Util

操作：int changeItemDBStatus（メディア種別, タイトル名, 業務種別, 会員名）

この中でファイルからデータを読み込み、修正し、書き込む。

readFile( ), writeFile( )：ファイルからの読み込みおよび書き込みの操作

■main 関数

コマンド行を読み込む。

コマンド行の内容を判定して RentalApp クラスの rent または back を呼び出す。

呼び出した結果でユーザに対してメッセージを出力する。

＊＊課題１１－２　　－オブジェクト指向設計の例―

オブジェクト指向らしく書き換える。オブジェクト指向とは「もの」を中心に設計すること。すなわち、貸出し、返却といった機能中心ではなく、「もの」であるアイテム（Video、CD）クラス、その上位クラスになる抽象的なアイテムクラスを作成する。
これは新規アイテム（例えばDVD）の追加する時に他アイテムへの影響を抑えるためである。さらに、業務（例えばアイテムの販売）が追加になったときに、他への影響が無いように「レンタルコントロール」クラスを設計する。このような方式は「デザインパターン」での「コントロールオブジェクト方式」という。

なお、１１－２の解答プログラムではデータベースへのアクセスは Util クラスに集中したが（１１－１に同じ)、この部分はデータベースをどう持つかにより変わってくる。

クラス　Util　　//　ユーティリティ 　DB 上のタイトルの状態（各モノの状態）を実際に書き換える関数 　ファイルへの読み書き関数を作成する。

クラス IItem　　//　アイテムの状態をセットする 　抽象クラス 　状態セット関数：ビデオ、CD 等モノに合わせてそれらの状態を変更する純粋仮想関数 　　実体はない 　コンストラクタ　　中味なし 　デストラクタ　　中味なし

クラス VideoItem　　//　クラス IItem の派生クラス 　メンバ変数　　タイトル名 　コンストラクタ　　タイトル名を初期化する 　状態セット関数：　//　純粋仮想関数の実体 　　状態（貸出中かどうか）と会員名を受け取り、ビデオである旨と 　　タイトル名、状態、会員名を Util の関数を呼び出すことで実現する。

クラス CDItem　　//　クラス IItem の派生クラス 　メンバ変数　　タイトル名 　コンストラクタ　　タイトル名を初期化する 　状態セット関数：　//　純粋仮想関数の実体 　　状態（貸出中かどうか）と会員名を受け取り、CD である旨と 　　タイトル名、状態、会員名を Util の関数を呼び出すことで実現する。

クラス　アイテム生成　　ItemFactory
　生成関数：アイテムの種類（ビデオか CD）とアイテムの名前（タイトル名）を
　　与えられ、ビデオならビデオのオブジェクトを 1 個生成する。
　　CD なら CD のオブジェクトを 1 個生成する。
　　いずれも、生成したオブジェクトへのポインタを返す。
　　（このポインタは仮想関数の修飾に使用される）

クラス　レンタルコントローラ　　RentalController
　貸出関数：
　　アイテムの種別、アイテムの名前（タイトル名)、会員名を与えられ、
　　生成関数を用いてアイテムを 1 個生成する。
　　　アイテムの種別、アイテムの名前（タイトル名）を渡す。
　　　生成したアイテムへのポインタが戻ってくるので、
　　　そのポインタで状態セット関数を呼び出す。
　　　状態セット関数へは貸出中（にする）と会員名を渡す。
　返却関数：
　　アイテムの種別、アイテムの名前（タイトル名)、会員名を与えられ、
　　生成関数を用いてアイテムを 1 個生成する。
　　　アイテムの種別、アイテムの名前（タイトル名）を渡す。
　　　生成したアイテムへのポインタが戻ってくるので、
　　　そのポインタで状態セット関数を呼び出す。
　　　状態セット関数へは返却（にする）と会員名を渡す。

メイン関数
　コマンドの引数からデータを取得する。
　　コマンド名　　（貸出　又は　返却）
　　アイテム種別　（ビデオ　又は　CD）
　　アイテムの名前　（タイトル名）
　　会員名
　レンタルコントローラを 1 個生成する。
　コマンド＝＝貸出　なら
　　レンタルコントローラの貸出関数を呼び出す。
　　　アイテム種別、アイテムの名前、会員名を渡す。
　コマンド＝＝返却　なら
　　レンタルコントローラの返却関数を呼び出す。
　　　アイテム種別、アイテムの名前、会員名を渡す。