応用プログラミング A　第 10 回演習問題　関数オーバーロード

下記の問題を解くプログラムを作成せよ。必須問題は授業時間中に必ず解答すること。

問題 1（必須）　オブジェクト作成と対応するコンストラクタの確認

オブジェクトの作成方法ごとに対応するコンストラクタ関数が必要なことを確認する。以下のリストと実行結果を元に、オブジェクトの宣言部分を完成させること。

```
#include <iostream>
using namespace std;

class myclass {
    int x;
public:
    myclass() {
        x = 0;
        cout << "初期化なしのコンストラクタ¥n";
    }
    myclass(int n) {
        x = n;
        cout << "初期化ありのコンストラクタ¥n";
    }
    int getx() { return x; }
};

int main() {
    myclass o1;    // 必要に応じて修正
    cout << "o1: " << o1.getx() << "¥n";
    myclass o2;    // 必要に応じて修正
    cout << "o2: " << o2.getx() << "¥n";
    myclass o3;    // 必要に応じて修正
    cout << "o3[0]: " << o3[0].getx() << "¥n";
    cout << "o3[1]: " << o3[1].getx() << "¥n";
    myclass o4;    // 必要に応じて修正
    cout << "o4[0]: " << o4[0].getx() << "¥n";
    cout << "o4[1]: " << o4[1].getx() << "¥n";
    myclass *o5;
    o5 = new myclass[2];
    if (!o5) {
        cout << "メモリ割り当てエラー¥n";
        return 1;
    }
    cout << "o5[0]: " << o5[0].getx() << "¥n";
    cout << "o5[1]: " << o5[1].getx() << "¥n";

    return 0;
}
```

```
初期化ありのコンストラクタ
o1: 0
初期化なしのコンストラクタ
o2: 0
初期化なしのコンストラクタ
初期化なしのコンストラクタ
o3[0]: 0
o3[1]: 0
初期化ありのコンストラクタ
初期化ありのコンストラクタ
o4[0]: 0
o4[1]: 0
初期化なしのコンストラクタ
初期化なしのコンストラクタ
o5[0]: 0
o5[1]: 0
続行するには何かキーを押してくだ
```

問題 2（必須）　コンストラクタ関数のオーバーロード

文字列を表す strtype クラスを作成する。このクラスは 2 つのコンストラクタを持つ。1 つは文字列を受け取ってその文字列をメモリに格納するコンストラクタ。もう 1 つは文字列と整数値を受け取って、整数値の数だけ繰り返された文字列をメモリに格納するコンストラクタである。以下のリストと実行結果を参考にこのコンストラクタの動作を確認するプログラムを完成させよ。なお、strtype クラスにリスト記載のメンバ以外を追加してはならない。

ヒント：仮引数を 1 つだけ受け取るコンストラクタは、例 5.2 の 2 の“「通常」のコンストラクタ”をそのまま使えばよい（“「通常」のコンストラクタ”の部分以外は使わないので全体をコピーしないように）。仮引数を 2 つ受け取るコンストラクタは“「通常」のコンストラクタ”を元に作成するが、メモリ割り当て成功後に先頭（*p か p[0]）にヌル文字'¥0'を代入し、strcpy()の代わりに strcat()を n 回繰り返す処理を記述する。確保するメモリの大きさは「文字数×繰り返し数＋１」に増えるので注意。表示の部分は練習問題 5.1 の 1 の解答を参考にするとよい。

```
class strtype {
    char *p;
public:
    strtype(char *s);
    strtype(char *s, int n);
    char *getstring() { return p; }
};

int main()
{
    strtype s1("Good!");
    strtype s2("Good!", 5);
```

```
s1: Good!
s2: Good!Good!Good!Good!Good!
続行するには何かキーを押してください . . .
```

問題 3　コピーコンストラクタ

問題 2 で作成したクラスにメモリを解放するデストラクタを追加する。そうすると、オブジェクトのコピーが作成されたときに望ましくない副作用が発生することがある。これを防ぐためには、コピーコンストラクタを作成する必要もある。さらに strtype クラスのオブジェクトと文字 c を受け取って、文字列中に含まれる文字 c の数を返す関数 ccount()を作成する。この関数を使って下図のような結果が得られるプログラムを作成せよ。

ヒント：文字列（char 型の配列）の先頭アドレスがポインタ p に代入されている場合、文字列の先頭文字が c と一致しているかは if (*p == c) で判定することができる。p をインクリメントすると次の要素を指すので、繰り返しの中で前述の if 文と p++;を書けばすべての文字を判定できる。ポインタを使わずに if (p[i] == c) のように配列の要素と 1 つずつ判定してもよい。

```
class strtype {
    char *p;
public:
    strtype(char *s);
    strtype(char *s, int n);
    strtype( ??? );  // コピーコンストラクタ
    ~strtype() { delete [] p; }
    char *getstring() { return p; }
};

int ccount(strtype x, char c)
{
    // 文字 c の数を数えて返す
}

int main()
{
    strtype s1("Good!");
    strtype s2("Good!", 5);

    cout << "s1 の o の数: " << ccount(s1, 'o') << "¥n";
    cout << "s2 の o の数: " << ccount(s2, 'o') << "¥n";
```

```
s1のoの数: 2
s2のoの数: 10
s1: Good!
s2: Good!Good!Good!Good!Good!
続行するには何かキーを押してく
```

問題 4　デフォルト引数

問題 2、問題 3 には仮引数を受け取る 2 種類のコンストラクタが存在するが、デフォルト引数を使用することによってコンストラクタは 1 つにすることができる。問 2 のプログラムをコンソールから文字列等を入力できるように変更し、下図のような結果が得られるプログラムを作成せよ。コンストラクタではデフォルト引数を利用すること。

```
class strtype {
    char *p;
public:
    strtype( ??? ); // デフォルト引数を利用
    strtype( ??? ); // コピーコンストラクタ
    ~strtype() { delete [] p; }
    char *getstring() { return p; }
};
```

下線部はコンソールからの入力

```
文字列s1を設定します。
文字列を入力してください: hello!
繰り返し文字列s2を設定します。
文字列を入力してください: help!
繰り返し回数を入力してください: 3
文字列s1とs2に含まれるある文字の数を数えます。
文字を入力してください: l
s1のlの数: 2
s2のlの数: 3
s1: hello!
s2: help!help!help!
続行するには何かキーを押してください . . .
```

問題 5（チャレンジ）　予定表

予定を表す Schedule クラスを作成する。このクラスは非公開のメンバ変数として、月を表す int 型 mon、日を表す int 型 day、予定を表す文字列の char 型*str を持つ。公開メンバとして、適当なコンストラクタとコピーコンストラクタとデストラクタ､値を設定する void set(int m, int d, char *s)、各値を読み出す int get_mon()、int get_day()、char *get_str()を持つ。set()関数では値の正当性をチェック（月は 1～12、日は月によって異なる）して不正な値なら設定せずにエラーメッセージを表示する。また、文字列がすでに格納されていたら一度そのメモリを解放してから再割り当てしたメモリに文字列をコピーするものとする。この Schedule クラスのオブジェクトを配列で作成し適当な値を入れる。そして、今日の予定が格納されているか判定する関数 bool today(Schedule ob)を作成し、オブジェクト配列の中に今日の予定が格納されている場合は予定を表示するプログラムを作成する。オブジェクトの値はプログラム中で適当に設定してよい。

```
今日の予定
        何もありません
全予定表示
4月4日　入学式
5月7日　創立記念日
6月20日 世界難民の日
7月17日 海の日（補講日）
8月11日 山の日
続行するには何かキーを押してください
```

今日の予定があればここに表示される

```
#include <iostream>
#include <ctime>
using namespace std;

class Schedule {
    int mon;
    int day;
    char *str;
public:
    Schedule() { mon = 0; day = 0; str = NULL; }
    Schedule(const Schedule &o);
    ~Schedule() { if (str) delete [] str; }
    void set(int m, int md, char *s);
    int get_mon() { return mon; }
    int get_day() { return day; }
    char *get_str() { return str; }
};

Schedule::Schedule(const Schedule &o) {
    // コピーコンストラクタの内容を記述
}

void Schedule::set(int m, int d, char *s) {
    int flag = 0;

    if (m < 1 || m > 12 || d < 1) {
        flag = 1;
    } else {
        switch(m) {
        case 1: case 3: case 5: case 7:
        case 8: case 10: case 12:
            if (d > 31) {
                flag = 1;
            }
            break;
        case 4: case 6: case 9: case 11:
            if (d > 30) {
                flag = 1;
            }
            break;
        case 2:
            if (d > 28) { // うるう年は考えない
                flag = 1;
            }
            break;
        }
    }
    if (flag) {
        cout << "日付が不正です¥n";
    } else {
        // 値の設定
    }
}

bool today(Schedule ob) {
    time_t now = time(NULL);
    struct tm *date = localtime(&now);

    if ( 日付チェックの式 )
        return true;
    else
        return false;
}
```