構造化からオブジェクト指向プログラミングへの考え方

１．非構造化プログラム

下記のような単純な繰り返し処理を例に、構造化とオブジェクト指向の概念を説明します。

```
#include "stdafx.h"
#include <iostream>
int _tmain(int argc, _TCHAR* argv[])
{
    using namespace std;
    cout << "処理１" << endl;
    cout << "処理２" << endl;
    cout << "処理３" << endl;

    cout << "処理１" << endl;
    cout << "処理２" << endl;
    cout << "処理３" << endl;

    getchar();
    return 0;
}
```

同じ処理を繰り返している

２．構造化プログラミング（順次　選択　繰り返し　の３大要素でプログラムを組む）

プログラム起動開始から順次処理を行い、共通する処理部分をくくり出して別の処理にまとめ、条件で分岐し、範囲を決めて繰り返す処理をいう。

構造化ステップ１：　共通部分（繰り返し）をくくり出す

```
#include "stdafx.h"
#include <iostream>

void  printout( void ){
    cout << "処理１" << endl;
    cout << "処理２" << endl;
    cout << "処理３" << endl;
}

int _tmain(int argc, _TCHAR* argv[])
{
    using namespace std;
    printout();
    printout();
    getchar();
    return 0;
}
```

同じ処理を関数化して、再利用しやすくする。

同じ命令を２回実行しているので、ループを使って、３回でも何回でも、要望に応じて変更しやすくする。

構造化ステップ２：　さらに共通部分（繰り返し）をくくり出して、処理を構造化する

```
#include "stdafx.h"
#include <iostream>

void    printout( void ){
      for( int i=1; i<=3; i++) { cout << "処理" << i << endl;}
}

int _tmain(int argc, _TCHAR* argv[])
{
      using namespace std;

      for( int i=1; i<=2; i++) { printout(); }

      getchar();
      return 0;
}
```

ここでは繰り返しループの中で、printout()という関数を実行しているが、メイン文では関数名を指定するだけにして簡素化している。
その詳細は、printout 関数を見れば良い。すなわちメイン関数の下に、printout 関数が配置されている構造になっている。

構造化すると for ループや別クラスを準備するなど、オーバーヘッドと呼ばれる処理が必要になり、プログラム行数が増える。しかし、たとえプログラムの行数が増えたとしても、「処理の流れを見やすくしよう」というのが、構造化プログラミングの基本である。

「各階層の機能が正しく動作していれば、全体も正しく機能する」と考えられるというのが、構造化プログラミングの基本的な考え方である。

３．オブジェクト指向プログラミング（隠蔽化　継承　多態性　の３大要素でプログラムを組む）

前節の構造化プログラムも見やすくなっているが、これだけでは他のプログラムから再利用するのが困難である。そこで再利用性を考慮して次のようにクラス(class)を利用して、プログラム同士を分離する。

オブジェクト指向化後：

```
#include "stdafx.h"
#include <iostream>

class PrintMachine{
  public:
      void    printout( void ){
            for( int i=1; i<=3; i++) { cout << "処理" << i << endl;}
      }
};

int _tmain(int argc, _TCHAR* argv[])
{
      using namespace std;
      PrintMachine myMachine;     //   クラスを利用するにはこのようにインスタンス変数を造る
      for( int i=1; i<=2; i++) { myMachine. printout( ); }
      getchar();
      return 0;
}
```

クラス化すると、メイン部分から、独立させる事ができる。

ここでは PrintMachine というクラスを作成して、これに printout()という機能を持たせている。別のクラスなので、様々なプログラムから、これをいつでも再利用できる。
ポイントとなるのは、オブジェクト指向プログラミングでは、処理を流れで考えるのではなく、もの（オブジェクト）の連携で考えることである。

従来の構造化プログラミングとオブジェクト指向プログラミングの違い
構造化プログラミング：(トップダウン型)
全体の流れを考えてから、個々の機能を詳細化してゆく方法
オブジェクト指向プログラミング：(ボトムアップ型)
必要な部品を先に考える方法

オブジェクト指向型開発の場合は全体からではなく、個々の部品（もの：オブジェクト）から作り、そのオブジェクトを組み合わせて全体を構成する。構造化プログラミングでは、まず全体が把握出来ないと設計も、プログラミングも出来ない。しかしオブジェクト指向プログラミングでは、全体を考える前に、部品化できるもの（オブジェクト）を探すことから始まる。

４．オブジェクト指向の考え方

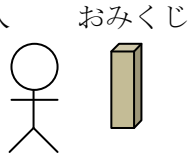
例として「おみくじ」システムを開発する。通常、おみくじの機械にはお金を入れるとおみくじが出てくる。

構造化プログラミング　で考えると次のようになる。
１）　プログラムを起動する
２）　乱数を元に「おみくじ」の結果を表示する

プログラム

```
main(){
  printout("吉");
}
```

オブジェクト指向プログラミング　　で考えると次のようになる。
１）　おみくじクラスと人間クラスの２つを考える
２）　人間プログラムを起動する
３）　人間クラスから、おみくじクラスを操作する
４）　おみくじクラスでは、乱数を元に「おみくじ」結果を人間クラスに表示する



処理の流れで考えず、この世界と同様に「もの」を操作することで目的を達成する。

５．練習　おみくじ表示システムを構造化プログラミングで作る

まずは以下のおみくじプログラムを理解して下さい。(簡単化のため、全て main メソッド内で処理を記述しています。)

プロジェクト名：　Omikuji1

以下のプログラムは、実行時の時間で乱数を発生させ、おみくじを表示させている。

```
#include "stdafx.h"
#include <iostream>
#include <time.h>
#include <windows.h>
#include <stdlib.h>   /*  乱数を発生させる srand rand  を使うために必要*/
int _tmain(int argc, _TCHAR* argv[])
{
      using namespace std;
      srand((unsigned int)time(NULL)); /*  乱数の初期値を時間から設定*/
      int result = rand() % 10 ; /* result  には 0 から 9 の乱数が代入される  */

      if      (result < 3) { cout << "  吉です" << endl;}
      else if(result < 7) { cout << "  中吉です" << endl;}
      else                      { cout << "  大吉です" << endl;}

      getchar();
      return 0;
}
```

これをオブジェクト指向にすると次のようになる。

```
#include "stdafx.h"
#include <iostream>
#include <time.h>
#include <windows.h>
#include <stdlib.h>   /*  乱数を発生させる srand rand  を使うために必要*/

class Omikuji{
   public:
   void print( void ){
      using namespace std;
      srand((unsigned int)time(NULL)); /*  乱数の初期値を時間から設定*/
      int result = rand() % 10 ; /* result  には 0 から 9 の乱数が代入される  */

      if      (result < 3) { cout << "  吉です" << endl;}
      else if(result < 7) { cout << "  中吉です" << endl;}
      else                      { cout << "  大吉です" << endl;}
   }
};

int _tmain(int argc, _TCHAR* argv[])
{
      Omikuji myunsei;   //Omikuji クラスから unsei という変数(実体)を作る
      myunsei.print();      //占いを実行する

      getchar();
      return 0;
}
```

課題　これを上書きして自分のオリジナルおみくじプログラムをオブジェクト指向で作りなさい。

以上