[prog2] Programming C++ (C6) Exercise Guide (Ex08)

11/02, Thursday 3rd period.

Ex08 について

- ジェネリックプログラミング

• プログラムが入力される変数型に依存しないで、正しく実行できるような実装をするスタイル

• 変数型が変わっても、実装したものが本質的に同じ構造となる要素を抽出して抽象的に表現する

- テンプレート関数

変数型（引数、戻り値）に依存せず、

想定される挙動があらゆる場合において正しく実行できる、抽象的な関数

内部の挙動は同じでも、扱う変数型が違う関数

- 例えば、ある入力された変数に＋5して返す関数

```
int func (int a){                  // int
  return a+5;
}

double func2 (double a){           // double
  return a+5;
}

char func3 (char a){               // char
  return a+5;
}
```

内部の挙動は同じでも、扱う変数型が違う関数

- このような関数たちを1つのプログラムの中で使いたいと思ったとき…

- 例えば、int 型を扱う関数に double型 の変数を渡すわけにもいかないし
- 型が違うものが必要なら、その分だけ関数を作る…？

  ⇒　過去（C言語）のスタイル

中身が　　return a+5;　　であることには変わりない

- この部分は一緒なんだし、どうにかならないか

template を使う

- テンプレートを使って、関数そのものを抽象化する

```
template <class T> T func (T a){
   return a + 5;
}
```

- **template <class T>** あるいは **template <typename T>** で抽象的に型を宣言する。（Tの部分は何でもよい）
続けて、Tを使って関数本体を書く。

※この場合、プロトタイプ宣言は
template <class T> T func(T);

template を使う

```
template <class T> T func (T a){
   return a + 5;
}
```

- このように書けば、引数と戻り値の型が同じであれば、どんな変数でも扱える関数となる
- ソースコードのコンパイル時に、書いたコード中の型を読み取って勝手に対応する関数を作成してくれる
- ただし、関数中に現れる演算が、引数に与えた変数型において定義されていなければエラーとなる
- int 型 の場合、 a = 10 なら 15 を返す（通常の加法）
- char 型の場合、a = 'C' なら 'H' を返す　(文字コードを5つ分進める)

※ この書き方で扱えるのは int func(int, int); や
double func(double, double)といった、型が全て同じであるものに限られる

template を使う（複数の型）

```
template <class T, class U> T func2 (T a, U b){
   return a * 5 + b;
}
```

- 異なる（かもしれない）型を扱う場合
- 抽象的な型が複数必要な場合は、,（カンマ）で並べる
- T と U で型が書き分けられていても、実際に渡す型は違っていてもよいし、同じであってもよい

○ 可能な呼び出し例：

```
int func(int, double);          string func(string, string);
```

× 不可能な呼び出し例：

```
int func(double, int);          string func(int, int);
↑           ↑    テンプレートに合わない型の配置はダメ
```