演習課題6

演習問題6-5(6-5)

演習問題6-2を継承して、以下を満たすクラス MedicalCheck3を作成しなさい

- 年齢を格納する変数、設定する関数、取得する関数を作る
- private
  - int age;
- public
  - void setAge(int a);
  - int getAge();

- 年齢に応じて、標準体重を求めるメンバ関数StandardBodyWeight()を作る
  - 15歳未満のとき、ローレル指数を基準にした標準体重を求める（MedicalCheck2::StandardBodyWeight()）
  - 15歳以上のとき、BMIを基準にした標準体重を求める（MedicalCheck::StandardBodyWeight()）

（次のスライドに続く）

演習課題6-5(続き)

```
int main() {
    int id, a;
    double h, w;

    cin >> id >> h >> w >> a;
    MedicalCheck3 mc;
    mc.setID(id);
    mc.setHeight(h);
    mc.setWeight(w);
    mc.setAge(a);
    cout << mc.BMI() << endl;
    cout << mc.rohrer() << endl;
    cout << mc.StandardBodyWeight() << endl;

    return 0;
}
```

main関数の例

```
1
1.61
65.2
14
25.1534
156.232
54.2527
```

```
2
1.61
65.2
15
25.1534
156.232
57.0262
```

演習問題6-6

- クラスInsuranceを定義する
  - private
    - int age; 年齢を格納
    - int color; 緑のとき0、青のとき1、ゴールドのとき2
  - Public
    - void setAge(int a); 年齢を設定
    - int getAge(); 年齢を取得
    - void setColor(int c); 免許の色を設定
    - int getColor(); 免許の色を取得
    - int fee(); 保険料を返す
      - 緑のとき 6000
      - 青のとき 7000
      - ゴールドのとき 5000
- (次に続く)

演習問題6-6続き

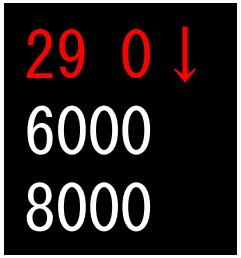
- クラスInsuranceを継承して、クラスInsurance2を定義する
  - public
    - int fee();　　保険料を返す（上書き定義）
      - 年齢と免許の色によって異なる
      - ヒント：　fee()関数内で、getAge(), getColor()して、年齢と色によって保険料を求める

年齢	～29	30～49	50～
青、緑	8000	6500	8500
ゴールド	5000	3500	4500

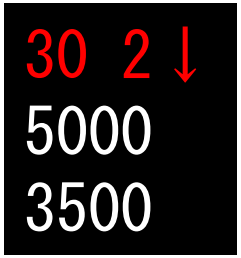
- (次に続く)

演習問題6-6(6-6)続き

```
int main()
{
    int a, c;

    cin >> a >> c;
    Insurance i1;
    i1.setAge(a);
    i1.setColor(c);
    cout << i1.fee() << endl;

    Insurance2 i2;
    i2.setAge(a);
    i2.setColor(c);
    cout << i2.fee() << endl;

    return 0;
}
```



実行例1



実行例2

演習課題6-7, 7-8のヒント

- 何度も出てきているので、もう書かなくても良いかもしれないが、queクラスは右のように定義される

```
class que {
private:
    int idx;
    int buf[10];
public:
    que()     {          idx = 0;    }
    void push(int v) {buf[idx++] = v;       }
    int pop()
    {
        int top = buf[0];
        for (int i = 1; i < idx; i++) {
            buf[i-1] = buf[i];
        }
        idx--;
        return top;
    }
};
```

ヒント：queクラス（基底クラス）

演習問題6-7

- プライオリティキューを作成しなさい
  - プライオリティキューとは順番を考慮したキューである
  - 今回の場合、小さい順に整列して値が取り出されるキューを指す
  - クラスqueを継承して、pqueを定義する
    - push()を上書き定義しても良いし、pop()を上書き定義しても良い
- （続く）

演習問題6-7

```
int main()
{
    que  q1;
    pque q2;
    int n;

    cin >> n;

    for (int i = 0; i < n; i++) {
        int v;
        cin >> v;
        q1.push(v);
        q2.push(v);
    }

    for (int i = 0; i < n; i++) {
        cout << q1.pop() << " ";
    }
    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << q2.pop() << " ";
    }
    cout << endl;
    return 0;
}
```

```
10
70 30 80 50 20 90 10 40 100 60
70 30 80 50 20 90 10 40 100 60
10 20 30 40 50 60 70 80 90 100
```

実行例1

演習問題6-8

- 演習問題6-2では、自然数しか入れられないスタックを作成した（同様のことをqueで行う）。
- クラスqueを継承して、以下の機能を持つque2を作りなさい
  - push()するときに、自然数かどうかチェックする
    - 自然数じゃなかったら、無視する
    - 値を格納するときは、上書き前のpush()を使う
  - 何個格納されているか分かるように、isEmpty()関数を定義する。
    - 予めqueクラスのidxを、privateからprotectedに移動しておく

（次のスライドに続く）

演習課題6-8(続き)

```
int main()
{
    int n;
    que  q1;
    que2 q2;

    cin >> n;
    for (int i = 0; i < n; i++) {
        int v;
        cin >> v;
        q1.push(v);
        q2.push(v);
    }

    for (int i = 0; i < n; i++) {
        cout << q1.pop() << " ";
    }
    cout << endl;

    while (q2.isEmpty() == false) {
        cout << q2.pop() << " ";
    }
    cout << endl;
    return 0;
}
```

main関数の例

```
class que {
protected:
    int buf[10];
protected:
    int idx;
public:
    que()
        :
```

ヒント：queクラスはidxを
protectedに移動しておく

```
10
-1 -2 -3 -4 5 -6 -7 -8 -9 -10
-1 -2 -3 -4 5 -6 -7 -8 -9 -10
5
```

実行例