練成問題

▪ **explicit** を付けて宣言されたコンストラクタは、 (1) コンストラクタと呼ばれ、単一の実引数でのオブジェクトの構築・初期化は (2) 。

▶ (2) の選択肢： (a) ( ) 形式のみに限定される　(b) = 形式のみに限定される
(c) ( ) 形式と = 形式の両方で行える

▪ コンストラクタやメンバ関数で、**new** 演算子によって確保した動的記憶域期間をもつ領域は、オブジェクトが破棄される際に自動的に解放 (3) 。

▶ 選択肢：(a)される　(b)されない
(c)されるかされないかは、オブジェクトの記憶域期間に依存する

▪ 代入演算子とコピーコンストラクタを定義する場合、自己代入・自己初期化に対する対処を行う必要がある。自己代入・自己初期化であるかどうかは、代入（初期化）元のオブジェクトへのポインタと、 (4) との等価性で判定する。

▪ 関数やクラスなどの部品では、エラー発生の際の対処を一意に決めることができない。というのも、エラーに対する対処は、部品の (5) 側で決められるべき場合が多いからである。 (6) のメカニズムを導入すると、エラーに対する対処の決定を、部品の (7) 側で柔軟に決定できるようになる。右に示すのは、エラーに対する処理を行うコードの形式の一例である。なお、“対処Ｃ”は、*ExpA* でも *ExpB* でもないエラーへの対処である。

▶ (5) と (7) の選択肢：(a)開発者　(b)利用者

```
(8) {
    // 処理
}
(9) (ExpA) {
    // 対処Ａ
}
(9) (ExpB) {
    // 対処Ｂ
}
(9) ( (10) ) {
    // 対処Ｃ
}
```

▪ クラス *C* のデストラクタの名前は (11) である。デストラクタに関する説明について、正しいものに○を、誤っているものに×を示せ。

(12) …オブジェクトの生存期間が尽きて破棄される直前に自動的に呼び出される。
(13) …任意の型と個数の引数を受け取ることができる。
(14) …任意の型を返却値型とすることができる。
(15) …多重定義できる。
(16) …**static** を付けて静的メンバ関数とすることができる。
(17) …定義を省略した場合は、コピーデストラクタがコンパイラによって自動的に提供される。

▪ 別のクラス内で定義されたクラスのことを、 (18) クラスと呼ぶ。クラス *C* 内で定義されたクラス *A* が公開属性をもっているとする。クラス *A* をアクセスする式は、クラス *C* の内部では (19) となり、クラス *C* の外部では (20) となる。

▪ 同一型のクラスオブジェクトの値によるオブジェクトの構築・初期化において、全データメンバを単純にコピーすべきでないクラスに対しては、 (21) を多重定義しなければならない。

▪ コンストラクタやメンバ関数で **new** 演算子によって確保した動的記憶域期間をもつ領域をオブジェクト破棄時に自動的に解放するためには、解放処理を (22) の中で行う。

なお、 (22) を定義しないクラスには、本体が空であって引数を受け取らない (23) (22) が、コンパイラによって自動的に定義される。

▪ 同一型のクラスオブジェクトどうしの代入において、全データメンバを単純にコピーすべきでないクラスに対しては、 (24) を多重定義しなければならない。

▪ 以下に示すのは、 (21) と (22) と (24) をもつクラスの典型的な定義例である。

```
class X {
    // 中略
public:
    X( (25) x) {
        if (this == (26) ) {
            // 自己初期化におけるデータメンバの値の適切な設定
        } else {
            // xの値をもとにしたデータメンバの値の適切な設定
        }
    }

    (27) () {
        // オブジェクトを破棄するための処理
    }

    (28) operator (29) ( (30) x) {
        if (this != (31) ) {
            // xの値をもとにしたデータメンバの値の適切な設定
        }
        return *this;
    }
};
```

▪ 例外を投げるのは、“ (32) 式”によって実現する。投げられた例外をつかまえることを“ (33) する”という。例外の (33) は、“ (34) ブロック”で行い、例外に対する対処は“例外 (35) ”によって行う。

▪ 右に示すのは、明示的コンストラクタをもつクラス *C* である。

```
class C {
public:
  (36) C(int) {
    // 中略
  }
};
```

▪ 前問のクラス *C* 型のオブジェクトの宣言として、正しいものに○を、誤っているものに×を埋めよ。

```
(37)  C a;
(38)  C b = 1;
(39)  C c(1);
(40)  C d = C(1);
```