C++言語

NO.4

目次

- （復習）キュー
  - プログラムの作り方
- コンストラクタ
  - コンストラクタを使って初期化する
  - 引数のあるコンストラクタ
- オブジェクトを返すメンバ関数

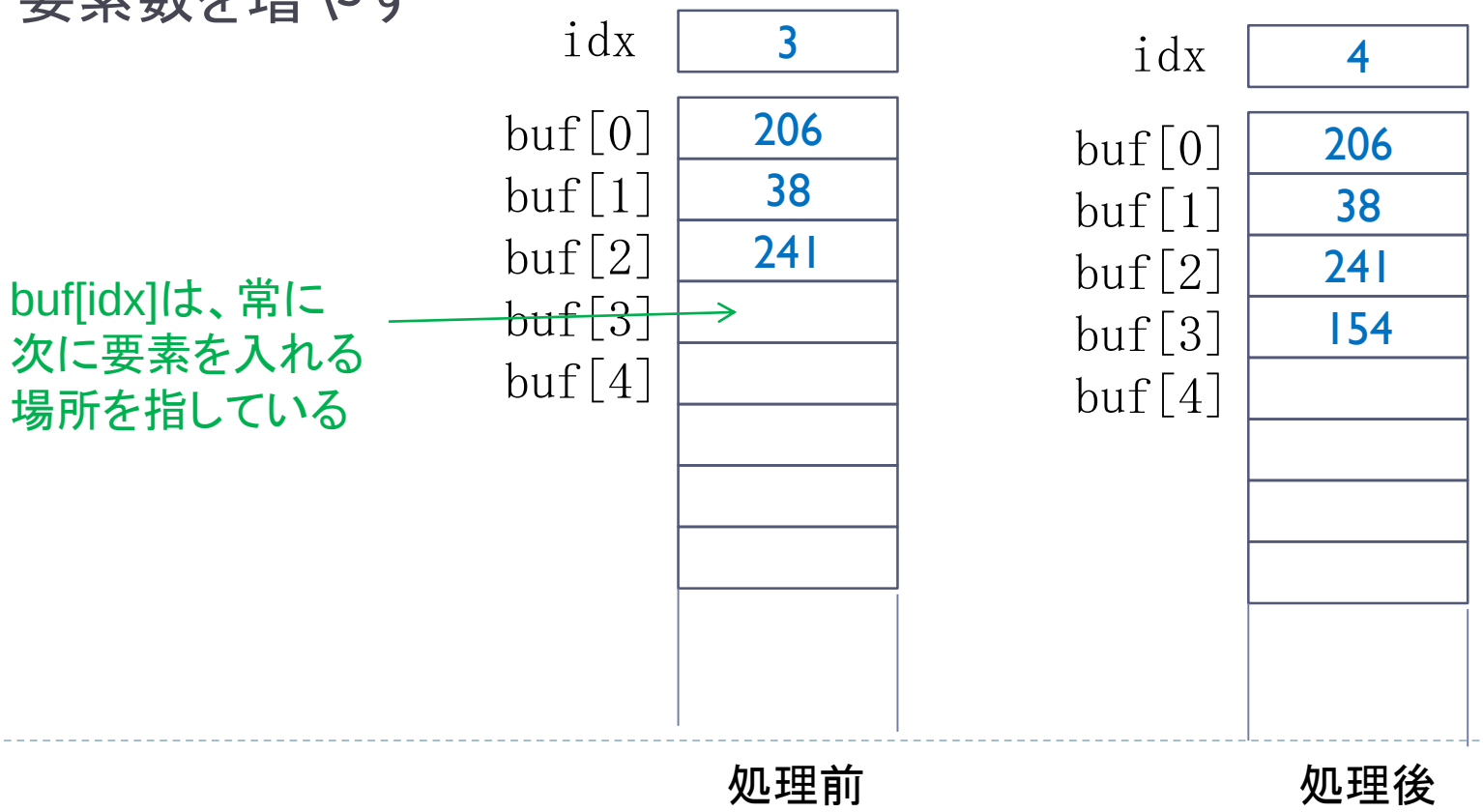
復習

スタックのプログラム

```
void push(int v)
{
    buf[idx] = v;
    idx++;
}
```

```
int pop
{
    idx--;
    return buf[idx];
}
```

- pushの動き
  - 値を入れる
    - どこに？
  - 要素数を増やす



キュー（再掲）

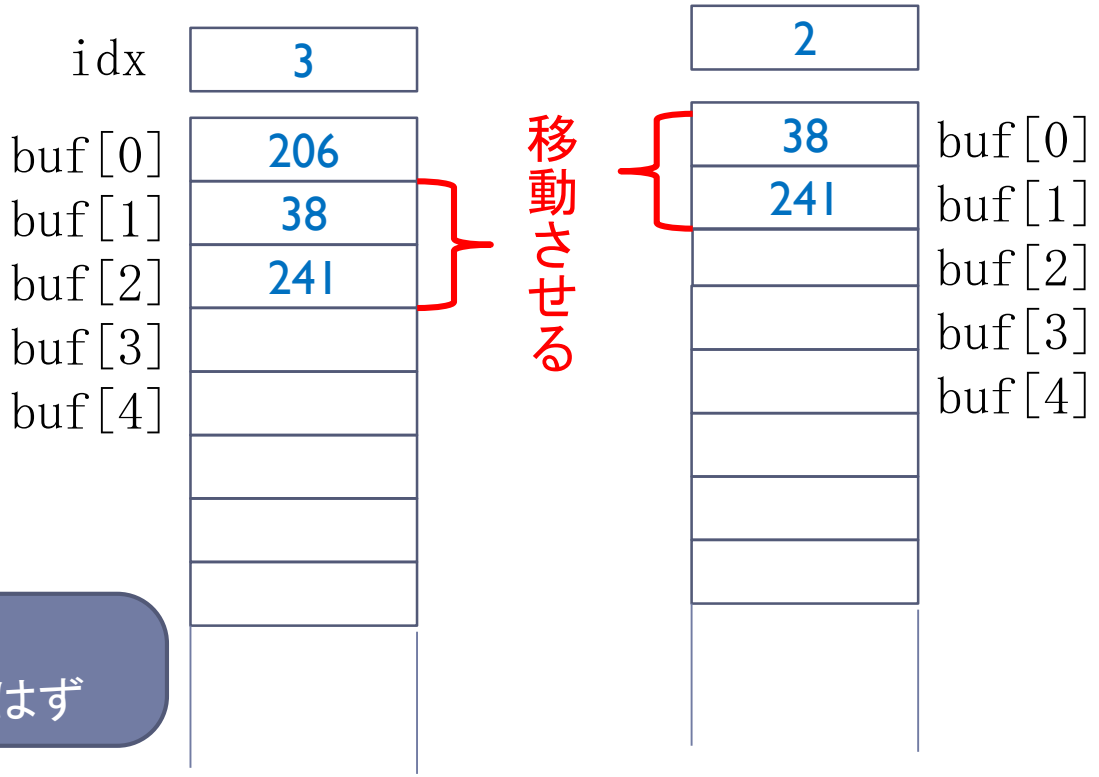
- キューの機能
  - キューを空にする（初期化）
  - 値を入れる
  - 値を取り出す
    - **先に**入れた値が，先に出てくる
- 制限
  - 話を簡単にするため，要素数を10個にする
  - エラーチェックはしない

キューの図解

キューのプログラム

【プログラムですること】
- 移動させる
- 先頭の値を返す
- 要素数を減らす

- pushは最後尾に詰める
  - スタックと同じ
- popは、配列の先頭を取り出す



演習問題2-7と
演習問題2-8ができるはず

コンストラクタ

初期値の代入（コンストラクタの必要性）

- 変数には初期値を入れる必要がある
  - 初期値設定は忘れることがある
- 初期値を自動で設定してくれるのがコンストラクタ
  - 特殊なメンバ関数

```
int main()
{
    stack a, b;

    a.empty();      // 初期化
    b.empty();      // 初期化
    a.push(100);
    b.push(90);
    a.push(80);
    b.push(70);
```

コンストラクタ

- オブジェクトが生成されるときに最初に呼び出されるメンバ関数
  - メンバ関数名は、クラス名と同じ
  - 返り値は無し（型指定も書かない）

```
class stack {
private:
    int idx;
    int buf[10];
public:
    void empty()
    {
        idx = 0;
    }
```

コンストラクタ使用前

```
class stack {
private:
    int idx;
    int buf[10];
public:
    stack()
    {
        idx = 0;
    }
```

コンストラクタ使用後

コンストラクタを使ったオブジェクトのmain関数

- main関数で初期化を意識しなくてよい
  - コンストラクタが自動で呼び出される

```
int main()
{
    stack a, b;

    a.empty();
    b.empty();
    a.push(100);
    b.push(90);
    a.push(80);
    b.push(70);
```

コンストラクタ使用前

```
int main()
{
    stack a, b;

    a.empty();
    b.empty();
    a.push(100);
    b.push(90);
    a.push(80);
    b.push(70);
```

コンストラクタ使用後

演習問題4-1(4-1)

- 演習問題3-2で作成したスタッククラスを、コンストラクタを使って書き直しなさい

```
4↓
10 20 30 40 50 60 70 80↓
70
80
50
60
30
40
10
20
```

改行も空白も入力の区切りになるので、入力を1行で書いて良い

```
2↓
11 22 33 44↓
33
44
11
22
```

```
int main()
{
    stack a, b;

    int n, v;

    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> v;       a.push(v);
        cin >> v;       b.push(v);
    }
    for (int i = 0; i < n; i++) {
        cout << a.pop() << endl;
        cout << b.pop() << endl;
    }
    return 0;
}
```

演習問題4-2(4-2)

- 演習問題3-3で作成したキュークラスを、コンストラクタを使って書き直しなさい

```
4↓
10 20 30 40 50 60 70 80↓
10
20
30
40
50
60
70
80
```

実行例

```
int main()
{
    que a, b;

    int n, v;

    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> v;    a.push(v);
        cin >> v;    b.push(v);
    }
    for (int i = 0; i < n; i++) {
        cout << a.pop() << endl;
        cout << b.pop() << endl;
    }
    return 0;
}
```

演習問題4-3(4-3)

- 演習問題3-5で作成したプログラムを以下のmain関数で書き直しなさい
- MovingObjectクラスを、コンストラクタを使って書き直しなさい
  - init()メンバ関数の機能をコンストラクタに持たせます。

```
int main()
{
    MovingObject a;
    int v;

    while ( 1 ) {
        cin >> v;
        if (v == 0) {
            break;
        }
        switch (v) {
            case 1:  a.turnLeft();   break;
            case 2:  a.step();       break;
            case 3:  a.turnRight();  break;
        }
    }
    int x, y;
    a.get(x, y);
    cout << "(" << x << "," << y << ")" << endl;
    return 0;
}
```

2 2 1 2 0↓
(-80,160)

引数のあるコンストラクタ

- コンストラクタに引数をつける
- オブジェクトを生成するときに引数をつける

```
class Date {
private:
    int year;
    int month;
    int day;
public:
    Date(int y, int m, int d) {
        year = y;
        month = m;
        day = d;
    }
};
```

Dateクラスの定義例
(コンストラクタの定義例)

```
int main()
{
    Date d(2014, 5, 9);
    return 0;
}
```

main関数

演習問題4-4(4-4)

- 演習問題3-4を引数のあるコンストラクタを使って書き換えなさい
  - set関数は削除しなくて良い
  - ただし、オブジェクト生成直後の、日付の設定はコンストラクタを使って行うこと
  - コンストラクタでは、不正な日付を排除する
    - 不正な日付のときは、0年0月0日に設定する
    - 返り値は返さない（返せない）
  - 実行例は右
  - main関数の例は次のページ

```
2014 1 32 ↓
正しく設定されていません
```

実行例1

```
2014 1 1 ↓
2014年1月1日が設定されました
次の日は2014年1月2日です
```

実行例2

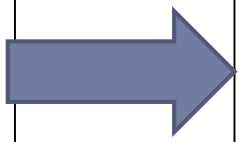
練習問題4-4(続き)

```
int main()
{
    int yy, mm, dd, y, m, d;

    cin >> yy >> mm >> dd;

    Date d1(yy, mm, dd);
    d1.get(y, m, d);
    if (y != 0 && m != 0 && d != 0) {
            cout << yy << "年" << mm << "月" << dd << "日が設定されました" << endl;
            d1.next();
            d1.get(yy, mm, dd);
            cout << "次の日は" << yy << "年" << mm << "月" << dd << "日です" << endl;
    } else {
            cout << "正しく設定されていません" << endl;
    }
    return 0;
}
```

デフォルトコンストラクタ（何もしないコンストラクタ）

- コンストラクタは必ず呼ばれる
- コンストラクタを定義していないときは、デフォルトコンストラクタが自動的に作られる
- デフォルトコンストラクタは、何もしないコンストラクタ

```
class Date {
    // コンストラクタの定義なし
};

int main()
{
   Date d;          // OK
}
```



```
class Date {
    Date(int y, int m, int d) {
        ….// コンストラクタ
    }
};

int main()
{
   Date d;                    // エラー
}
```

引数つきコンストラクタを追加するとエラーになる

複数のコンストラクタを定義できる（オーバーロード）

```
class Date {
    Date(int y, int m, int d) {
        ….// コンストラクタ
    }
    Date() {
        year = month = day = 1;
    }
};

int main()
{
   Date d1;
   Date d2(2014, 4, 31);

}
```

メンバ関数の引数に、オブジェクトを渡す

メンバ関数の引数にオブジェクトを渡す

- メンバ変数を参照するためには、メンバ変数名を書く
  - 今までと同じ
- 引数で渡されたオブジェクト内の、メンバ変数を参照するには、「オブジェクト名.メンバ変数名」とする
  - （例）「d.month」のようにする

```
class Date {
protected:
    int year;
    int month;
    int day;
public:
    Date(int y, int m, int d) {
        year = y;
        month = m;
        day = d;
    }
    int sub(Date d) {
        year -= d.year;
        month -= d.month;
        day -= d.day;
    }
};
```

Dateクラスの定義例
(コンストラクタの使用例)

演習問題4-5(4-5)

- 演習問題4-4で作成したプログラムに以下のメンバ関数を付け加えなさい
- 日付の差を求めるメンバ関数sub()加える
  - ただし適切に動くようにすること
  - うるう年は考慮しなくて良い
  - 引数で渡される方が、古い日付であることを仮定して良い
  - ヒント：同じに日付になるまで、next()を呼び出して、その回数を数えればよい

```
2014 8 31↓
2014 9 1↓
2014年8月31日から2014年9月1日までは1日です
```

実行例

  - main関数の例は次のページ

演習課題 4-5(続き)

```
int main()
{
    int yy, mm, dd;

    cin >> yy >> mm >> dd;
    Date d1(yy, mm, dd);
    d1.get(yy, mm, dd);
    if (yy == 0 && mm == 0 && dd == 0) {
            cout << "正しく設定されていません" << endl;
            return 1;
    }
    cin >> yy >> mm >> dd;
    Date d2(yy, mm, dd);
    d2.get(yy, mm, dd);
    if (yy == 0 && mm == 0 && dd == 0) {
            cout << "正しく設定されていません" << endl;
            return 1;
    }
    d1.get(yy, mm, dd);
    cout << yy << "年" << mm << "月" << dd << "日から";
    d2.get(yy, mm, dd);
    cout << yy << "年" << mm << "月" << dd << "日までは";
    cout << d2.sub(d1) << "日です" << endl;
    return 0;
}
```

main関数の例

演習問題4-6(4-6)

- コンピュータ業界では、四角形のことを矩形（くけい）と呼ぶことが多い
  - 「矩形」の漢字に注意。「短」じゃない
- 矩形を格納するクラスを作成しなさい
  - 今回扱う矩形は、XY軸に平行な形をした正方形であることを仮定する
  - 以下を格納するメンバ変数を持っている
    - 四角形の左上の座標（x, y)
      - どちらもdouble型
    - 辺の長さ: double w

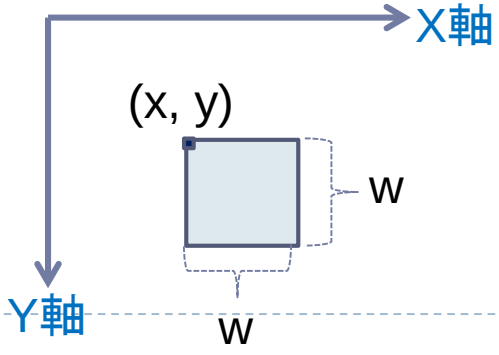
演習問題4-6（続き）

```
7  7  10
17  17  10
aとbは矩形が接しています
bとcは矩形が接しています
aとbは内接円が接しています
```

実行例

- 以下のメンバ関数を持っている
  - コンストラクタ（引数なし）
    - x = 0, y = 0, w = 10が格納される
  - コンストラクタ（引数あり）
    - (x, y), wの初期値を受け取る
  - isOverlapR()関数
    - 引数に与えられた矩形の内接円と、自分自身の矩形に内接する円との、2個の円が重なっているか判定する
      - 内接円の中心(x + w/2, y + w/2)
      - 内接円の半径 w/2
      - それぞれの内接円の中心間の距離が、内接円の半径r1とr2の和より短いとき、接している

赤丸が接しているかを調べる。
（左、接している。右、接していない）

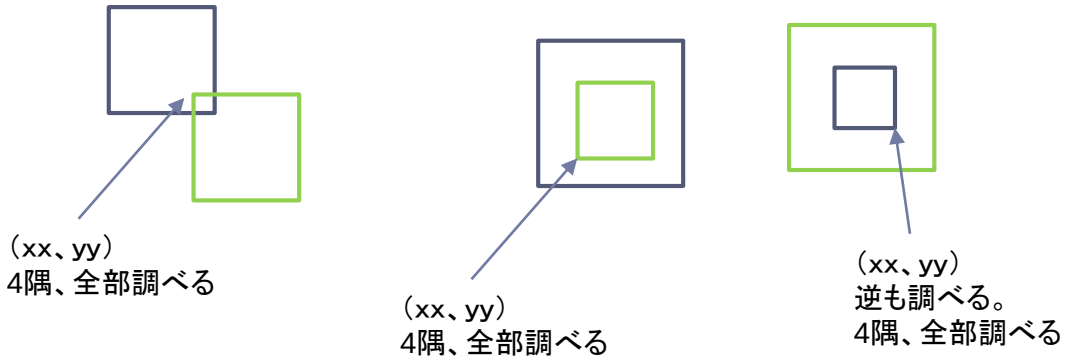


※Y軸の向きに注意

(続く)

演習問題4-6（続き）

- 以下のメンバ関数を持っている
  - isOverlap()関数
    - 引数に与えられた矩形が、自分自身と重なっているか判定する
    - ヒントなので、この通りに実装しなくても良い
      - 与えられた座標(xx, yy)が、矩形の中にあるかどうかを調べるメソッドを用意する
      - 上記のメソッドを使って、重なっているか調べたい矩形の四隅の座標が、矩形内にあるか調べる。矩形内にあれば、接している
      - 逆も調べる。すなわち、重なっているか調べたい矩形の中に、自分の四隅の座標が矩形内にあるか調べる。矩形内にあれば、接している。
      - そうでないときは、接していない



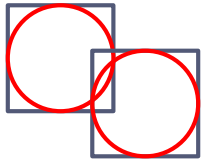
(続く)

演習問題4-6（続き）

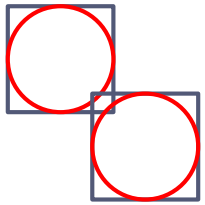
```
0 0 10
5 5 8
-10 -10 20
```

テストセットの5個目

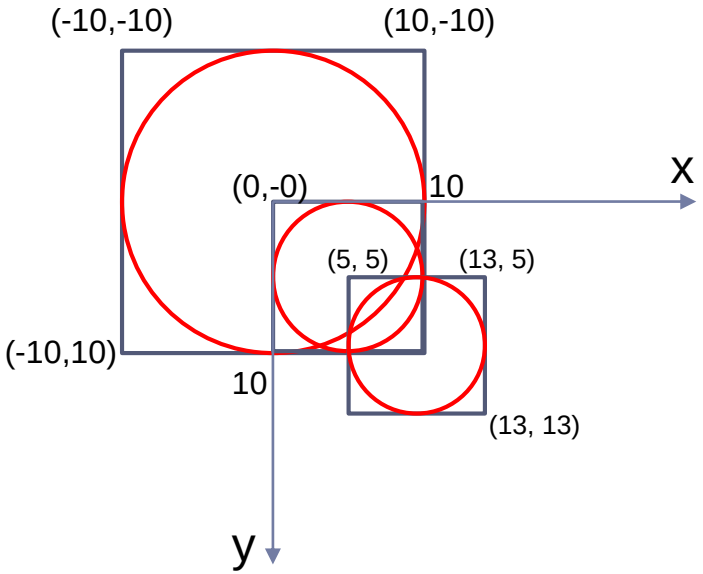
- 実行のイメージ



矩形も内接円も重なっている



矩形は重なっている
内接円は重なっていない



- main関数の例は次のページ

演習課題4-6 (続き)

```
int main()
{
    Kukei a;

    double x, y, w;
    cin >> x >> y >> w;
    Kukei b(x, y, w);
    cin >> x >> y >> w;
    Kukei c(x, y, w);


    if (a.isOverlap(b)) {
            cout << "aとbは矩形が接しています" << endl;
    }
    if (b.isOverlap(c)) {
            cout << "bとcは矩形が接しています" << endl;
    }
    if (c.isOverlap(a)) {
            cout << "cとaは矩形が接しています" << endl;
    }
    if (a.isOverlapR(b)) {
            cout << "aとbは内接円が接しています" << endl;
    }
    if (b.isOverlapR(c)) {
            cout << "bとcは内接円が接しています" << endl;
    }
    if (c.isOverlapR(a)) {
            cout << "cとaは内接円が接しています" << endl;
    }
    return 0;
}
```

main関数の例

計算結果にオブジェクトを返す

オブジェクトを返す

```
class Date2 {
private:
    int year;
    int month;
    int day;
public:
    Date2(int y, int m, int d) {
        year = y;
        month = m;
        day = d;
    }
    Date2 get() {
        return Date2(year, month, day + 1);
    }
    Date2 get2() {
        Date2 a(year, month, day + 1);
        return a;
    }
};
```

返り値の型

オブジェクトを作成

- 返り値をオブジェクトにする
  - 型宣言
- return文にオブジェクトを指定
  - get()では、return文内でコンストラクタを使って作成