

listクラス

- vectorと同様コンテナクラスの一つ

`list<型名> インスタンス名{ 初期値 }`

- それぞれの要素が次の要素へのポインタで接続されていて、要素の削除や追加が容易
(前期のC言語で学習したリスト構造を実現)
- 配列とは異なり、添え字番号を使って
要素へ直接アクセスすることはできない

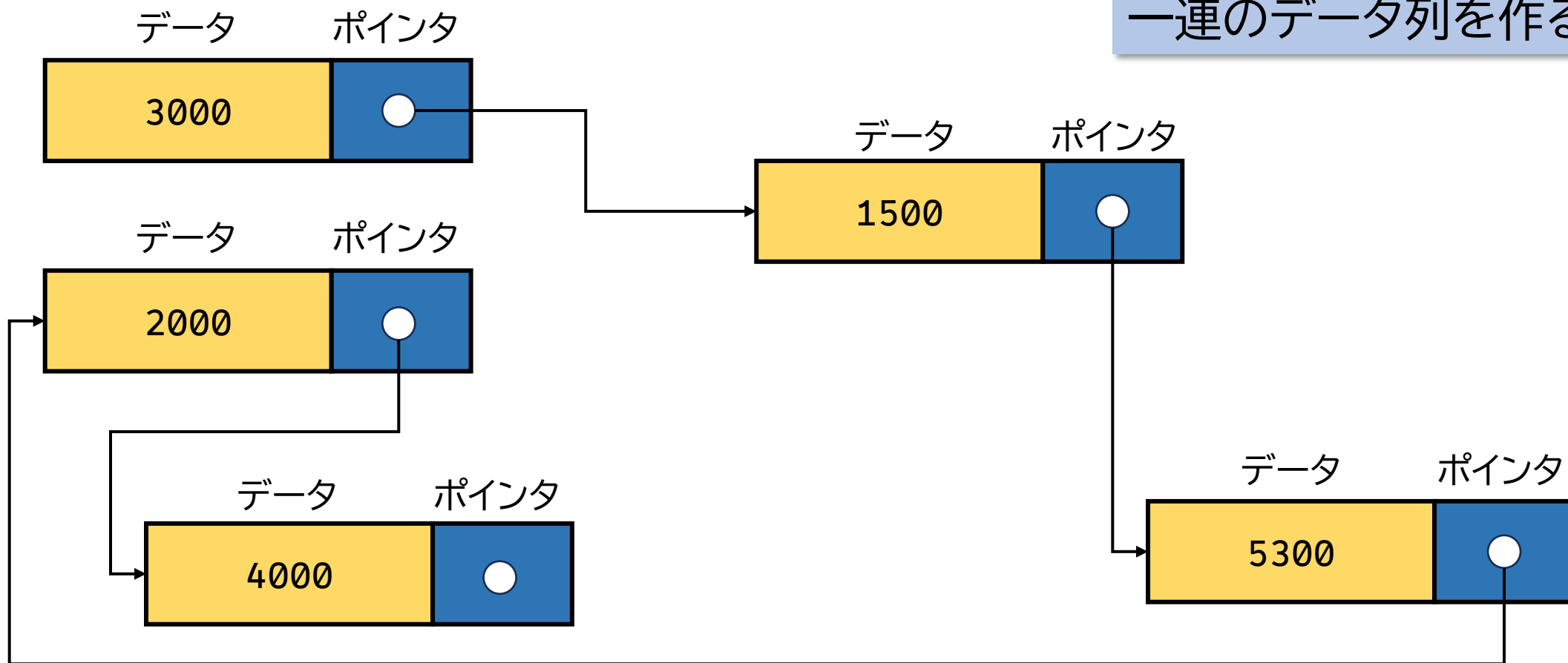
listクラス

データの並び

3000 → 1500 → 5300 → 2000 → 4000

• リスト構造

次のデータのアドレスを
ポインタとして保持して
一連のデータ列を作る



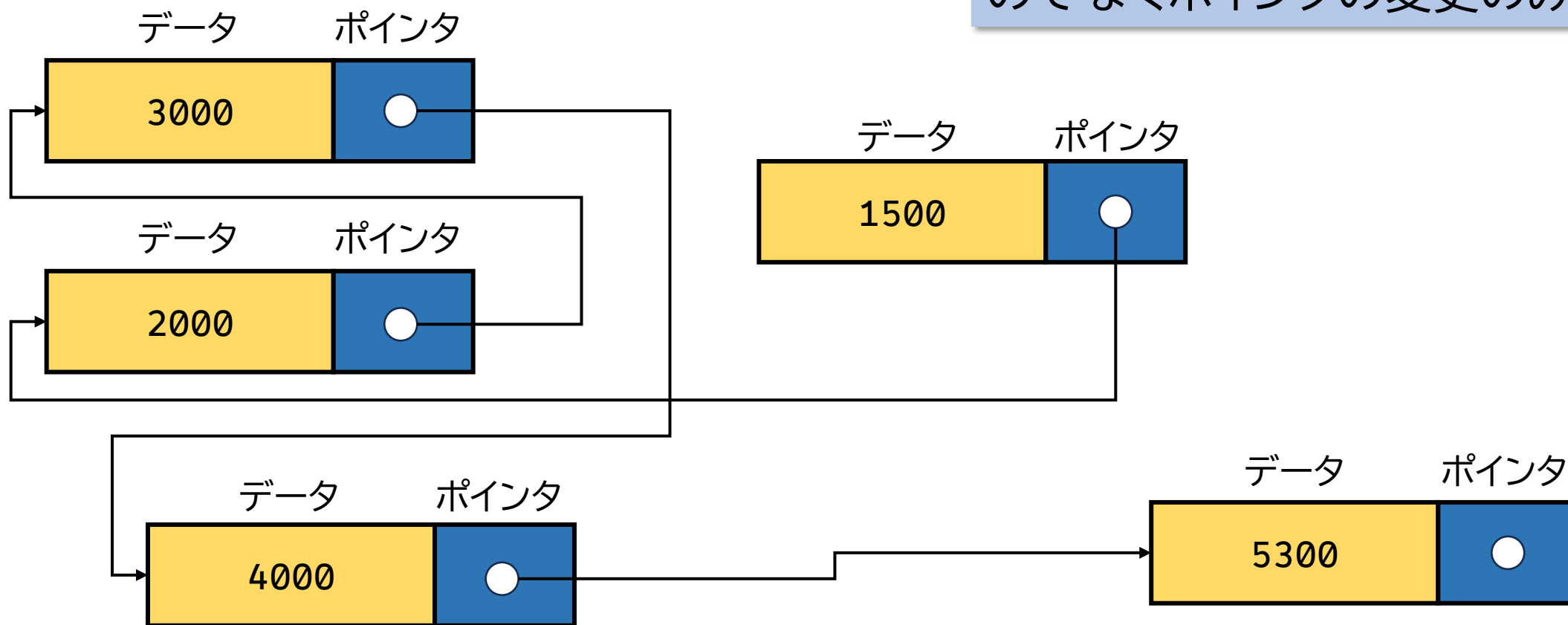
listクラス

データの並び

1500 → 2000 → 3000 → 4000 → 5300

• リスト構造(ソート後)

昇順でのソートを実施
(データを並び変えている
のではなくポインタの変更のみ)



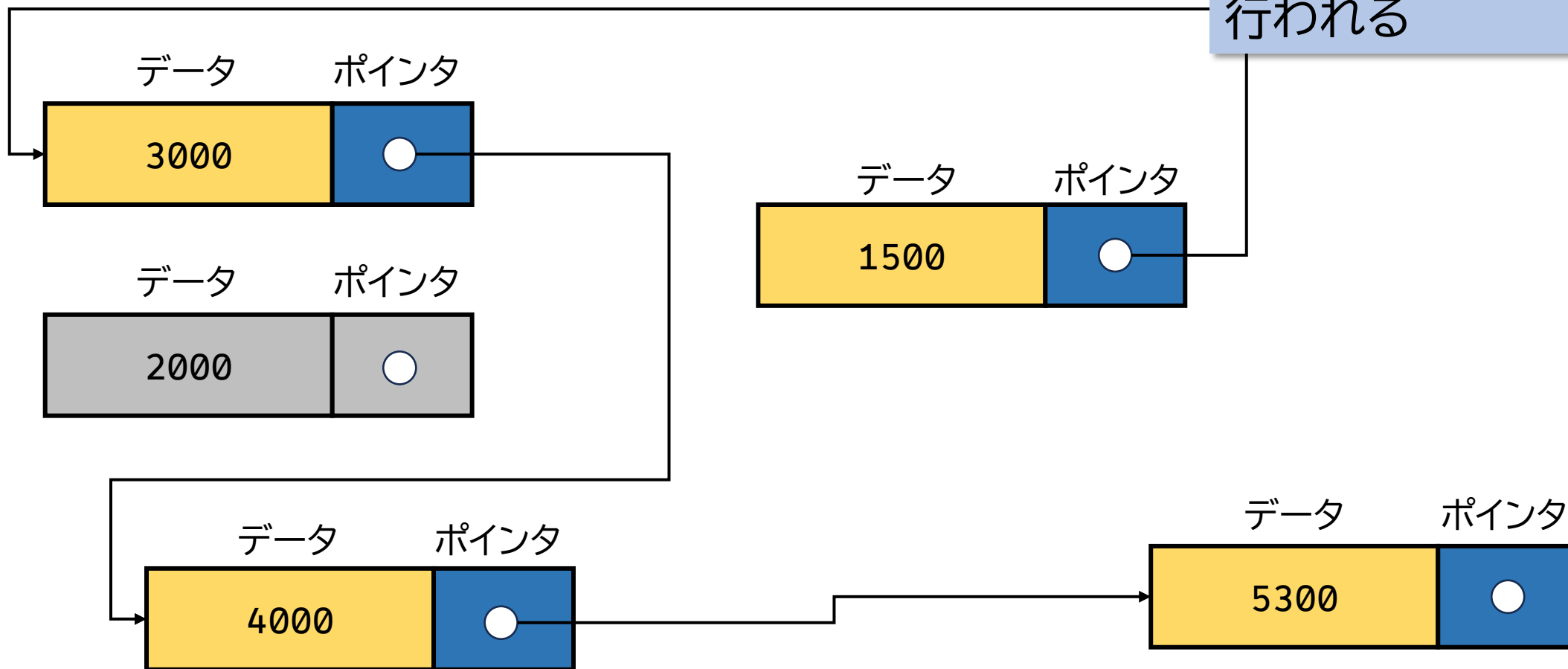
listクラス

データの並び remove(2000)

1500 → 3000 → 4000 → 5300

• リスト構造(2000を削除)

データを削除すると
ポインタの付け替えが
行われる



listクラス

- listクラスのメンバ関数

- `size()` : リストの全データ数をカウント
- `push_back()` : 末尾にデータを付け加える
- `push_front()` : 先頭にデータを付け加える
- `pop_back()` : 末尾のデータを削除
- `pop_front()` : 先頭のデータを削除
- `erase()` : イテレータ指定場所のデータを削除
- `insert()` : イテレータ指定場所へデータを追加
- `remove()` : 引数で指定した値を削除
- `unique()` : リスト内の重複データを削除
- `sort()` : データ列を昇順で並び替える
- `reverse()` : データ列を逆並びにする

listクラス

- 教科書P231~232 Sample606
- C++作業フォルダ内にSample606フォルダを作成
`mkdir Sample606`
`cd Sample606`
- main.cppを作成
`copy nul main.cpp`


listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{};
    li.push_back(1);    //データ末尾への追加[1]
    li.push_back(2);    //データ末尾への追加[1 2]
    li.push_front(3);   //データ先頭への追加[3 1 2]
    auto itr = li.begin(); //イテレータをリストの先頭にする
    itr++;               //イテレータをひとつ進める
    li.insert(itr, 4);    //イテレータの位置へ挿入[3 4 1 2]
    for (itr = li.begin(); itr != li.end(); itr++) {
        cout << *itr << " "; //イテレータを使って内容を表示
    }
    cout << endl;
    return 0;
}
```

listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
     list<int> li{5, 8, 2}; //初期値の設定[5 8 2]
    li.push_back(1);      //[5 8 2 1]
    li.push_back(2);      //[5 8 2 1 2]
    li.push_front(3);     //[3 5 8 2 1 2]
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);     //[3 4 5 8 2 1 2]
    for (itr = li.begin(); itr != li.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;
    return 0;
}
```


listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort(); //昇順ソートの実行
    for (itr = li.begin(); itr != li.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;
    return 0;
}
```

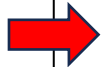
//[3 4 5 8 2 1 2]
[1 2 2 3 4 5 8]



listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort();
    li.unique(); //重複データの削除    [1 2 3 4 5 8]
    for (itr = li.begin(); itr != li.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;
    return 0;
}
```



listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort();
    li.unique();
    for (auto& d : li) {
        cout << d << " ";
    }
    cout << endl;
    return 0;
}
```

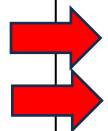
//範囲for文への書き換え

listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort();
    li.unique();
    for (auto d : li ) {
        cout << d << " ";
    }
    cout << endl;
    return 0;
}
```

//範囲for文への書き換え



listクラス

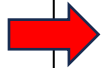
main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort();
    li.unique();
    li.remove(8); //データの値が 8 のものをリストから削除 [1 2 3 4 5]
    for (auto d : li) {
        cout << d << " ";
    }
    cout << endl;
```

listクラス

main.cpp (Sample606)

```
#include <list>
#include <iostream>
using namespace std;
int main() {
    list<int> li{5, 8, 2};
    li.push_back(1);
    li.push_back(2);
    li.push_front(3);
    auto itr = li.begin();
    itr++;
    li.insert(itr, 4);
    li.sort();
    li.unique();
    li.remove(3);
    li.reverse(); //データ列を逆転させる [5 4 3 2 1]
    for (auto d : li) {
        cout << d << " ";
    }
}
```



listクラス

- listまとめ

- リスト構造を実現するコンテナクラス
- 配列のように添え字での直接アクセスは不可
- データはイテレータを使ってアクセス可能
- メンバ関数によって、ソートや重複削除等、さまざまな処理が可能
- リストに対して処理を自分でする場合は、ピンポイントでデータを探すことができないので、リスト全体からデータをイテレータで検索して処理する必要がある