

## Note

- All Codes of this assignment are available [here](#)

## Question

Dining Philosophers Algorithm implemented using Monitors

### Code

```
/*
 * @author YashKumarVerma / 19BCE2669
 * @for: Lab Digital Assignment 2
 * @description: Implementation of Dining Philosophers Problem using
Monitors
 */

#include<iostream>
#include<vector>

using namespace std;

/** Utility function to format display as requireds **/
class console{
public:
    static void log(string status, string message){
        cout << "[" << status << "] :: " << message << endl;
    }

    static void log(string status, int message){
        cout << "[" << status << "] :: " << message << endl;
    }
};

class Philosopher {
public:
    bool left;
    bool right;
    bool takenDinner;

    Philosopher(bool left = false, bool right = false){
        this->left = left;
        this->right = right;
        this->takenDinner = false;
    }

    bool justTakenTwoForks(){
        return this->left && this->right;
    }
}
```

```

        bool hasLeftAndWantsRight () {
            return this->left && !this->right;
        }

        bool hasNoForks () {
            return !this->left && !this->right;
        }
    };

/***
 * @class Fork
 * @description To handle all operating related to fork / chopstick
 ***/

class Fork {
private:
    bool taken;
public:
    Fork(bool taken = false) {
        this->taken = taken;
    }

    bool isTaken () {
        return this->taken;
    }

    void take () {
        this->taken = true;
    }

    void release () {
        this->taken = false;
    }
};

/***
 * @class DiningPhilosophers
 * @description To handle all operating related to problem
 ***/
class DiningPhilosophers{
private:
    vector<Philosopher> philosopher;
    vector<Fork> fork;

public:
    int completedPhilosophers;
    int totalPhilosophers;

    /** set initial conditions **/
    DiningPhilosophers(int totalPhilosophers = 5) {
        this->completedPhilosophers = 0;
        this->totalPhilosophers = totalPhilosophers;

        for(int i=0; i<this->totalPhilosophers; i++) {

```

```

        this->philosopher.push_back(Philosopher());
        this->fork.push_back(Fork());
    }

}

/** start dinner by philosophers */
void startDinner() {
    console::log("starting", "meal");
    while(this->completedPhilosophers < this->totalPhilosophers) {
        console::log("totalPhilosophers", this->totalPhilosophers);
        console::log("completedPhilosophers", this-
>completedPhilosophers);

        for(int philosopherId=0; philosopherId< this-
>totalPhilosophers; philosopherId += 1) {
            console::log("starting dinner", philosopherId);
            this->attemptToEat(philosopherId);
        }
    }
    console::log("completing", "meal");
}

/** make attempts to eat */
void attemptToEat(int philosopherId) {
    /** if dinner already taken */
    if(this->philosopher[philosopherId].takenDinner) {
        console::log("taken dinner", philosopherId);
    }

    /** then the philosopher can take dinner */
    else if(this->philosopher[philosopherId].justTakenTwoForks()) {
        console::log("taken dinner", philosopherId);
        this->philosopher[philosopherId].takenDinner = true;

        int otherFork = philosopherId - 1;
        if(otherFork == -1) {
            otherFork = this->totalPhilosophers - 1;
        }

        /** releasing forks */
        this->fork[philosopherId].release();
        this->fork[otherFork].release();

        console::log("philosopher", philosopherId );
        console::log("released fork", philosopherId );
        console::log("released fork", otherFork );

        this->completedPhilosophers += 1;
    }

    else if(this->philosopher[philosopherId].hasLeftAndWantsRight()) {
        if(philosopherId == (this->totalPhilosophers - 1)) {
            if(!this->fork[philosopherId].isTaken()) {
                this->fork[philosopherId].take();
            }
        }
    }
}

```

```

        this->philosopher[philosopherId].right = true;
        console::log("philosopher",philosopherId );
        console::log("picks fork", philosopherId );
    }
    else{
        console::log("philosopher",philosopherId );
        console::log("waiting for fork", philosopherId );
    }
}
/** case of last philosopher ***/
else {
    int duplicatePhilosopherId = philosopherId;
    philosopherId -= 1;

    if(philosopherId == -1) {
        philosopherId = this->totalPhilosophers - 1;
    }

    if(!this->fork[philosopherId].isTaken()) {
        this->fork[philosopherId].take();
        this->philosopher[duplicatePhilosopherId].right = true;
        console::log("philosopher",philosopherId );
        console::log("picks fork", duplicatePhilosopherId );
    }
    else{
        console::log("philosopher",duplicatePhilosopherId );
        console::log("waiting for fork", philosopherId );
    }
}
}

else if(this->philosopher[philosopherId].hasNoForks()) {
    if(philosopherId == this->totalPhilosophers - 1) {
        if(!this->fork[philosopherId-1].isTaken()) {
            this->fork[philosopherId-1].take();
            this->philosopher[philosopherId].left = true;
            console::log("philosopher",philosopherId );
            console::log("picks fork", philosopherId-1);
        }
        else {
            console::log("philosopher",philosopherId );
            console::log("waiting for fork", philosopherId-1);
        }
    }
    /** case of last philosopher ***/
    else {
        if(!this->fork[philosopherId].isTaken()) {
            this->fork[philosopherId].take();
            this->philosopher[philosopherId].left = true;
            console::log("philosopher",philosopherId );
            console::log("picks fork", philosopherId);
        }
        else{
            console::log("philosopher",philosopherId );
            console::log("waiting for fork", philosopherId);
        }
    }
}

```

```
        }
    }
}
};

/** Driver code ***/
int main() {
    DiningPhilosophers diningPhilosophers = DiningPhilosophers(5);
    diningPhilosophers.startDinner();
}
```

## Code Preview

yash@hephaestus: ~  
bat DiningPhilosopherMonitors.cpp

```
File: DiningPhilosopherMonitors.cpp
you, a few seconds ago | 1 author (you)
1  /*** code                                     ## Note
2  * @author YashKumarVerma / 19BCE2669          = All Codes of this assignment are available [here](https://github.com/
3  * @for: Lab Digital Assignment 2
4  * @description: Implementation of Dining Philosophers Problem using Monitors[semester3]
5  */#include
6
7 #include<iostream>                                ## Question
8 #include<vector>                                 Dining Philosophers Algorithm implemented using Monitors
9
10 using namespace std;
11
12 /** Utility function to format display as requireds **/## Code
13 class _console{
14 public:
15     static void log(string status, string message){
16         cout << "[" << status << "] :: " << message << endl;
17     }
18
19     static void log(string status, int message){
20         cout << "[" << status << "] :: " << message << endl;
21     }
22 };
23
24
25 class Philosopher {
26 public:
27     bool left;
28     bool right;
29     bool takenDinner;
30
31     Philosopher(bool left = false, bool right = false){
32         this->left = left;
33         this->right = right;
34         this->takenDinner = false;
35     }
36
37     void eat();
38
39     void think();
40
41     void sleep();
42
43     void takeDinner();
44
45     void releaseDinner();
46
47     void printStatus();
48
49     void printMessage(string message);
50
51     void printError(string error);
52
53     void printWarning(string warning);
54
55     void printInfo(string info);
56
57     void printDebug(string debug);
58
59     void printTrace(string trace);
60
61     void printLog(string log);
62
63     void printFatal(string fatal);
64
65     void printErrorFatal(string errorFatal);
66
67     void printWarningFatal(string warningFatal);
68
69     void printInfoFatal(string infoFatal);
70
71     void printDebugFatal(string debugFatal);
72
73     void printTraceFatal(string traceFatal);
74
75     void printLogFatal(string logFatal);
76
77     void printFatalFatal(string fatalFatal);
78
79     void printErrorFatalFatal(string errorFatalFatal);
80
81     void printWarningFatalFatal(string warningFatalFatal);
82
83     void printInfoFatalFatal(string infoFatalFatal);
84
85     void printDebugFatalFatal(string debugFatalFatal);
86
87     void printTraceFatalFatal(string traceFatalFatal);
88
89     void printLogFatalFatal(string logFatalFatal);
90
91     void printFatalFatalFatal(string fatalFatalFatal);
92
93     void printErrorFatalFatalFatal(string errorFatalFatalFatal);
94
95     void printWarningFatalFatalFatal(string warningFatalFatalFatal);
96
97     void printInfoFatalFatalFatal(string infoFatalFatalFatal);
98
99     void printDebugFatalFatalFatal(string debugFatalFatalFatal);
100    void printTraceFatalFatalFatal(string traceFatalFatalFatal);
101
102    void printLogFatalFatalFatal(string logFatalFatalFatal);
103
104    void printFatalFatalFatalFatal(string fatalFatalFatalFatal);
105
106    void printErrorFatalFatalFatalFatal(string errorFatalFatalFatalFatal);
107
108    void printWarningFatalFatalFatalFatal(string warningFatalFatalFatalFatal);
109
110    void printInfoFatalFatalFatalFatal(string infoFatalFatalFatalFatal);
111
112    void printDebugFatalFatalFatalFatal(string debugFatalFatalFatalFatal);
113
114    void printTraceFatalFatalFatalFatal(string traceFatalFatalFatalFatal);
115
116    void printLogFatalFatalFatalFatal(string logFatalFatalFatalFatal);
117
118    void printFatalFatalFatalFatalFatal(string fatalFatalFatalFatalFatal);
119
120    void printErrorFatalFatalFatalFatalFatal(string errorFatalFatalFatalFatalFatal);
121
122    void printWarningFatalFatalFatalFatalFatal(string warningFatalFatalFatalFatalFatal);
123
124    void printInfoFatalFatalFatalFatalFatal(string infoFatalFatalFatalFatalFatal);
125
126    void printDebugFatalFatalFatalFatalFatal(string debugFatalFatalFatalFatalFatal);
127
128    void printTraceFatalFatalFatalFatalFatal(string traceFatalFatalFatalFatalFatal);
129
130    void printLogFatalFatalFatalFatalFatal(string logFatalFatalFatalFatalFatal);
131
132    void printFatalFatalFatalFatalFatalFatal(string fatalFatalFatalFatalFatalFatal);
133
134    void printErrorFatalFatalFatalFatalFatalFatal(string errorFatalFatalFatalFatalFatalFatal);
135
136    void printWarningFatalFatalFatalFatalFatalFatal(string warningFatalFatalFatalFatalFatalFatal);
137
138    void printInfoFatalFatalFatalFatalFatalFatal(string infoFatalFatalFatalFatalFatalFatal);
139
140    void printDebugFatalFatalFatalFatalFatalFatal(string debugFatalFatalFatalFatalFatalFatal);
141
142    void printTraceFatalFatalFatalFatalFatalFatal(string traceFatalFatalFatalFatalFatalFatal);
143
144    void printLogFatalFatalFatalFatalFatalFatal(string logFatalFatalFatalFatalFatalFatal);
145
146    void printFatalFatalFatalFatalFatalFatalFatal(string fatalFatalFatalFatalFatalFatalFatal);
147
148    void printErrorFatalFatalFatalFatalFatalFatalFatal(string errorFatalFatalFatalFatalFatalFatalFatal);
149
150    void printWarningFatalFatalFatalFatalFatalFatalFatal(string warningFatalFatalFatalFatalFatalFatalFatal);
151
152    void printInfoFatalFatalFatalFatalFatalFatalFatal(string infoFatalFatalFatalFatalFatalFatalFatal);
153
154    void printDebugFatalFatalFatalFatalFatalFatalFatal(string debugFatalFatalFatalFatalFatalFatalFatal);
155
156    void printTraceFatalFatalFatalFatalFatalFatalFatal(string traceFatalFatalFatalFatalFatalFatalFatal);
157
158    void printLogFatalFatalFatalFatalFatalFatalFatal(string logFatalFatalFatalFatalFatalFatalFatal);
159
160    void printFatalFatalFatalFatalFatalFatalFatalFatal(string fatalFatalFatalFatalFatalFatalFatalFatal);
161
162    void printErrorFatalFatalFatalFatalFatalFatalFatalFatal(string errorFatalFatalFatalFatalFatalFatalFatal);
163
164    void printWarningFatalFatalFatalFatalFatalFatalFatal(string warningFatalFatalFatalFatalFatalFatal);
165
166    void printInfoFatalFatalFatalFatalFatalFatal(string infoFatalFatalFatalFatalFatal);
167
168    void printDebugFatalFatalFatalFatalFatal(string debugFatalFatalFatalFatal);
169
170    void printTraceFatalFatalFatalFatal(string traceFatalFatalFatal);
171
172    void printLogFatalFatalFatal(string logFatalFatal);
173
174    void printFatalFatal(string fatalFatal);
175
176    void printErrorFatal(string errorFatal);
177
178    void printWarningFatal(string warningFatal);
179
180    void printInfoFatal(string infoFatal);
181
182    void printDebugFatal(string debugFatal);
183
184    void printTraceFatal(string traceFatal);
185
186    void printLogFatal(string logFatal);
187
188    void printFatalFatalFatal(string fatalFatalFatal);
189
190    void printErrorFatalFatal(string errorFatalFatal);
191
192    void printWarningFatalFatal(string warningFatalFatal);
193
194    void printInfoFatalFatal(string infoFatalFatal);
195
196    void printDebugFatalFatal(string debugFatalFatal);
197
198    void printTraceFatalFatal(string traceFatalFatal);
199
200    void printLogFatalFatal(string logFatalFatal);
201
202    void printFatalFatalFatalFatal(string fatalFatalFatalFatal);
203
204    void printErrorFatalFatalFatal(string errorFatalFatalFatal);
205
206    void printWarningFatalFatalFatal(string warningFatalFatalFatal);
207
208    void printInfoFatalFatalFatal(string infoFatalFatalFatal);
209
210    void printDebugFatalFatalFatal(string debugFatalFatalFatal);
211
212    void printTraceFatalFatalFatal(string traceFatalFatalFatal);
213
214    void printLogFatalFatalFatal(string logFatalFatalFatal);
215
216    void printFatalFatalFatalFatalFatal(string fatalFatalFatalFatalFatal);
217
218    void printErrorFatalFatalFatalFatal(string errorFatalFatalFatalFatal);
219
220    void printWarningFatalFatalFatalFatal(string warningFatalFatalFatalFatal);
221
222    void printInfoFatalFatalFatalFatal(string infoFatalFatalFatal);
223
224    void printDebugFatalFatalFatal(string debugFatalFatal);
225
226    void printTraceFatalFatal(string traceFatal);
227
228    void printLogFatal(string logFatal);
229
230    void printFatalFatalFatalFatalFatal(string fatalFatalFatalFatal);
231
232    void printErrorFatalFatalFatalFatal(string errorFatalFatalFatal);
233
234    void printWarningFatalFatalFatalFatal(string warningFatalFatal);
235
236    void printInfoFatalFatalFatal(string infoFatalFatal);
237
238    void printDebugFatalFatal(string debugFatal);
239
240    void printTraceFatal(string traceFatal);
241
242    void printLogFatal(string logFatal);
243
244    void printFatalFatalFatalFatalFatalFatal(string fatalFatalFatalFatal);
245
246    void printErrorFatalFatalFatalFatal(string errorFatalFatalFatal);
247
248    void printWarningFatalFatalFatalFatal(string warningFatalFatal);
249
250    void printInfoFatalFatalFatal(string infoFatalFatal);
251
252    void printDebugFatalFatal(string debugFatal);
253
254    void printTraceFatal(string traceFatal);
255
256    void printLogFatal(string logFatal);
257
258    void printFatalFatalFatalFatalFatalFatalFatal(string fatalFatalFatalFatal);
259
260    void printErrorFatalFatalFatalFatal(string errorFatalFatal);
261
262    void printWarningFatalFatalFatalFatal(string warningFatal);
263
264    void printInfoFatalFatalFatal(string infoFatal);
265
266    void printDebugFatalFatal(string debugFatal);
267
268    void printTraceFatal(string traceFatal);
269
270    void printLogFatal(string logFatal);
271
272    void printFatalFatalFatalFatalFatalFatalFatalFatal(string fatalFatal);
273
274    void printErrorFatalFatalFatalFatal(string errorFatal);
275
276    void printWarningFatalFatalFatal(string warningFatal);
277
278    void printInfoFatalFatal(string infoFatal);
279
280    void printDebugFatal(string debugFatal);
281
282    void printTraceFatal(string traceFatal);
283
284    void printLogFatal(string logFatal);
285
286    void printFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
287
288    void printErrorFatalFatalFatalFatal(string error);
289
290    void printWarningFatalFatalFatal(string warning);
291
292    void printInfoFatalFatal(string info);
293
294    void printDebugFatal(string debug);
295
296    void printTraceFatal(string trace);
297
298    void printLogFatal(string log);
299
300    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
301
302    void printErrorFatalFatalFatalFatal(string error);
303
304    void printWarningFatalFatalFatal(string warning);
305
306    void printInfoFatalFatal(string info);
307
308    void printDebugFatal(string debug);
309
310    void printTraceFatal(string trace);
311
312    void printLogFatal(string log);
313
314    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
315
316    void printErrorFatalFatalFatalFatal(string error);
317
318    void printWarningFatalFatalFatal(string warning);
319
320    void printInfoFatalFatal(string info);
321
322    void printDebugFatal(string debug);
323
324    void printTraceFatal(string trace);
325
326    void printLogFatal(string log);
327
328    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
329
330    void printErrorFatalFatalFatalFatal(string error);
331
332    void printWarningFatalFatalFatal(string warning);
333
334    void printInfoFatalFatal(string info);
335
336    void printDebugFatal(string debug);
337
338    void printTraceFatal(string trace);
339
340    void printLogFatal(string log);
341
342    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
343
344    void printErrorFatalFatalFatalFatal(string error);
345
346    void printWarningFatalFatalFatal(string warning);
347
348    void printInfoFatalFatal(string info);
349
350    void printDebugFatal(string debug);
351
352    void printTraceFatal(string trace);
353
354    void printLogFatal(string log);
355
356    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
357
358    void printErrorFatalFatalFatalFatal(string error);
359
360    void printWarningFatalFatalFatal(string warning);
361
362    void printInfoFatalFatal(string info);
363
364    void printDebugFatal(string debug);
365
366    void printTraceFatal(string trace);
367
368    void printLogFatal(string log);
369
370    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
371
372    void printErrorFatalFatalFatalFatal(string error);
373
374    void printWarningFatalFatalFatal(string warning);
375
376    void printInfoFatalFatal(string info);
377
378    void printDebugFatal(string debug);
379
380    void printTraceFatal(string trace);
381
382    void printLogFatal(string log);
383
384    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
385
386    void printErrorFatalFatalFatalFatal(string error);
387
388    void printWarningFatalFatalFatal(string warning);
389
390    void printInfoFatalFatal(string info);
391
392    void printDebugFatal(string debug);
393
394    void printTraceFatal(string trace);
395
396    void printLogFatal(string log);
397
398    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
399
400    void printErrorFatalFatalFatalFatal(string error);
401
402    void printWarningFatalFatalFatal(string warning);
403
404    void printInfoFatalFatal(string info);
405
406    void printDebugFatal(string debug);
407
408    void printTraceFatal(string trace);
409
410    void printLogFatal(string log);
411
412    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
413
414    void printErrorFatalFatalFatalFatal(string error);
415
416    void printWarningFatalFatalFatal(string warning);
417
418    void printInfoFatalFatal(string info);
419
420    void printDebugFatal(string debug);
421
422    void printTraceFatal(string trace);
423
424    void printLogFatal(string log);
425
426    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
427
428    void printErrorFatalFatalFatalFatal(string error);
429
430    void printWarningFatalFatalFatal(string warning);
431
432    void printInfoFatalFatal(string info);
433
434    void printDebugFatal(string debug);
435
436    void printTraceFatal(string trace);
437
438    void printLogFatal(string log);
439
440    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
441
442    void printErrorFatalFatalFatalFatal(string error);
443
444    void printWarningFatalFatalFatal(string warning);
445
446    void printInfoFatalFatal(string info);
447
448    void printDebugFatal(string debug);
449
450    void printTraceFatal(string trace);
451
452    void printLogFatal(string log);
453
454    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
455
456    void printErrorFatalFatalFatalFatal(string error);
457
458    void printWarningFatalFatalFatal(string warning);
459
460    void printInfoFatalFatal(string info);
461
462    void printDebugFatal(string debug);
463
464    void printTraceFatal(string trace);
465
466    void printLogFatal(string log);
467
468    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
469
470    void printErrorFatalFatalFatalFatal(string error);
471
472    void printWarningFatalFatalFatal(string warning);
473
474    void printInfoFatalFatal(string info);
475
476    void printDebugFatal(string debug);
477
478    void printTraceFatal(string trace);
479
480    void printLogFatal(string log);
481
482    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
483
484    void printErrorFatalFatalFatalFatal(string error);
485
486    void printWarningFatalFatalFatal(string warning);
487
488    void printInfoFatalFatal(string info);
489
490    void printDebugFatal(string debug);
491
492    void printTraceFatal(string trace);
493
494    void printLogFatal(string log);
495
496    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
497
498    void printErrorFatalFatalFatalFatal(string error);
499
500    void printWarningFatalFatalFatal(string warning);
501
502    void printInfoFatalFatal(string info);
503
504    void printDebugFatal(string debug);
505
506    void printTraceFatal(string trace);
507
508    void printLogFatal(string log);
509
510    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
511
512    void printErrorFatalFatalFatalFatal(string error);
513
514    void printWarningFatalFatalFatal(string warning);
515
516    void printInfoFatalFatal(string info);
517
518    void printDebugFatal(string debug);
519
520    void printTraceFatal(string trace);
521
522    void printLogFatal(string log);
523
524    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
525
526    void printErrorFatalFatalFatalFatal(string error);
527
528    void printWarningFatalFatalFatal(string warning);
529
530    void printInfoFatalFatal(string info);
531
532    void printDebugFatal(string debug);
533
534    void printTraceFatal(string trace);
535
536    void printLogFatal(string log);
537
538    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
539
540    void printErrorFatalFatalFatalFatal(string error);
541
542    void printWarningFatalFatalFatal(string warning);
543
544    void printInfoFatalFatal(string info);
545
546    void printDebugFatal(string debug);
547
548    void printTraceFatal(string trace);
549
550    void printLogFatal(string log);
551
552    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
553
554    void printErrorFatalFatalFatalFatal(string error);
555
556    void printWarningFatalFatalFatal(string warning);
557
558    void printInfoFatalFatal(string info);
559
560    void printDebugFatal(string debug);
561
562    void printTraceFatal(string trace);
563
564    void printLogFatal(string log);
565
566    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
567
568    void printErrorFatalFatalFatalFatal(string error);
569
570    void printWarningFatalFatalFatal(string warning);
571
572    void printInfoFatalFatal(string info);
573
574    void printDebugFatal(string debug);
575
576    void printTraceFatal(string trace);
577
578    void printLogFatal(string log);
579
580    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
581
582    void printErrorFatalFatalFatalFatal(string error);
583
584    void printWarningFatalFatalFatal(string warning);
585
586    void printInfoFatalFatal(string info);
587
588    void printDebugFatal(string debug);
589
590    void printTraceFatal(string trace);
591
592    void printLogFatal(string log);
593
594    void printFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatalFatal(string fatal);
595
596    void printErrorFatalFatalFatalFatal(string error);
597
598    void printWarningFatalFatalFatal(string warning);
599
599 }
```

```
yash@hephaestus: ~
bat DiningPhilosopherMonitors.cpp
33     this->right = right;
34     this->takenDinner = false;
35 }
36
37 bool justTakenTwoForks(){
38     return this->left && this->right;
39 }
40
41 bool hasLeftAndWantsRight(){
42     return this->left && !this->right;      ## Question
43 }
44
45 bool hasNoForks(){
46     return !this->left && !this->right;      ##### Code
47 }
48
49 */
50
51 /* @class Fork
52 * @description To handle all operating related to fork / chopstick
53 */ 198CE2669-DA3.md
54
55 class Fork { Algorithm.cpp
56     private: DiningPhilosopherMonitors.cpp
57         bool taken;
58     public: DiningSemaphore.cpp
59     Fork(bool taken = false){ ReaderAndWriterSemaphore.cpp
60         this->taken = taken;
61     }
62     bool isTaken(){ ReaderAndWriterSemaphore.cpp
63         return this->taken;
64     }
65     void take(){ ReaderAndWriterSemaphore.cpp
66         this->taken = true;
67     }
68
69 MySQL>
```

yash@hephaestus: ~

bat DiningPhilosopherMonitors.cpp

```
void take(){
    this->taken = true;
}
void release(){
    this->taken = false;
};

class DiningPhilosophers{
private:
    vector<Philosopher> philosopher;
    vector<Fork> fork;

public:
    int completedPhilosophers;
    int totalPhilosophers;

    DiningPhilosophers(int totalPhilosophers = 5){
        this->completedPhilosophers = 0;
        this->totalPhilosophers = totalPhilosophers;

        for(int i=0; i<this->totalPhilosophers; i++){
            this->philosopher.push_back(Philosopher());
            this->fork.push_back(Fork());
        }
    }

    void startDinner(){
        console::log("starting", "meal");
        while(this->completedPhilosophers < this->totalPhilosophers){
            console::log("totalPhilosophers", this->totalPhilosophers);
        }
    }
};
```

yash@hephaestus: ~

bat DiningPhilosopherMonitors.cpp

```
102     console::log("starting", "meal");
103     while(this->completedPhilosophers < this->totalPhilosophers){// cannot determine recent change or authors
104         console::log("totalPhilosophers", this->totalPhilosophers);
105         console::log("completedPhilosophers", this->completedPhilosophers);
106
107         for(int philosopherId=0; philosopherId< this->totalPhilosophers; philosopherId += 1){
108             console::log("starting dinner", philosopherId);
109             this->attemptToEat(philosopherId);
110         }
111     }## Question
112     console::log("completing", "meal");Dining Philosophers Algorithm Implemented using Monitors
113 }
114
115 /* make attempts to eat */
116 void attemptToEat(int philosopherId){#### Code
117     /* if dinner already taken */
118     if(this->philosopher[philosopherId].takenDinner){
119         console::log("taken dinner", philosopherId);
120     }
121 #### Code Preview
122     /* then the philosopher can take dinner */
123     else if(this->philosopher[philosopherId].justTakenTwoForks()){
124         console::log("taken dinner", philosopherId);
125         this->philosopher[philosopherId].takenDinner = true;
126     }
127     int otherFork = philosopherId + 1;
128     if(otherFork == -1){
129         otherFork = this->totalPhilosophers - 1;
130     }
131     ReaderWriterSemaphore::acquire();
132     /* releasing forks */
133     notes->fork[philosopherId].release();
134     notes->fork[otherFork].release();
135
136     console::log("philosopher", philosopherId );
137     console::log("released fork", philosopherId );
138     console::log("released fork", otherFork );
139 }
```

```

yash@hephaestus: ~
bat DiningPhilosopherMonitors.cpp

137 //FOAM-NOTES
138     console::log("released fork", philosopherId );
139     console::log("released fork", otherFork ); // you, a few seconds ago | 1 author (You)
140     this->completedPhilosophers += 1; // Note
141 }
142
143 else if(this->philosopher[philosopherId].hasLeftAndWantsRight()){
144     if(philosopherId == (this->totalPhilosophers - 1)){
145         if(!this->fork[philosopherId].isTaken()){
146             this->fork[philosopherId].take();
147             this->philosopher[philosopherId].right = true;
148             console::log("philosopher",philosopherId );
149             console::log("picks fork", philosopherId );
150         }
151     }
152     else{ //## Code
153         console::log("philosopher",philosopherId );
154         console::log("waiting for fork", philosopherId );
155     }
156 }
157 /** case of last philosopher ***/
158 else { //## Code Preview
159     int duplicatePhilosopherId = philosopherId;
160     philosopherId -= 1;
161
162     if(philosopherId == -1){
163         philosopherId = this->totalPhilosophers - 1;
164     }
165
166     if(!this->fork[philosopherId].isTaken()){
167         this->fork[philosopherId].take();
168         this->philosopher[duplicatePhilosopherId].right = true;
169         console::log("philosopher",philosopherId );
170         console::log("picks fork", duplicatePhilosopherId );
171     }
172     else{
173         console::log("philosopher",duplicatePhilosopherId );
174         console::log("waiting for fork", philosopherId );
175     }
176 }
177
178 else if(this->philosopher[philosopherId].hasNoForks()){
179     if(philosopherId == this->totalPhilosophers - 1){
180         if(!this->fork[philosopherId-1].isTaken()){
181             this->fork[philosopherId-1].take();
182             this->philosopher[philosopherId].left = true;
183             console::log("philosopher",philosopherId );
184             console::log("picks fork", philosopherId );
185         }
186     }
187     else {
188         console::log("philosopher",philosopherId );
189         console::log("waiting for fork", philosopherId-1 );
190     }
191 }
192 /** case of last philosopher ***/
193 else {
194     if(!this->fork[philosopherId].isTaken()){
195         this->fork[philosopherId].take();
196         this->philosopher[philosopherId].left = true;
197         console::log("philosopher",philosopherId );
198         console::log("picks fork", philosopherId );
199     }
200     else{
201         console::log("philosopher",philosopherId );
202         console::log("waiting for fork", philosopherId );
203     }
204 }
205
206 /**
207 */

```

```
yash@hephaestus: ~
bat DiningPhilosopherMonitors.cpp

175 }
176 }
177 } else if(this->philosopher[philosopherId].hasNoForks()){
178     if(philosopherId == this->totalPhilosophers - 1){
179         if(!this->fork[philosopherId-1].isTaken()){
180             this->fork[philosopherId-1].take();
181             this->philosopher[philosopherId].left = true;
182             console::log("philosopher",philosopherId );
183             console::log("picks fork", philosopherId-1);
184         }
185     } else {
186         console::log("philosopher",philosopherId );
187         console::log("waiting for fork", philosopherId-1);
188     }
189 }
190 /** case of last philosopher */
191 else {
192     if(!this->fork[philosopherId].isTaken()){
193         this->fork[philosopherId].take();
194         this->philosopher[philosopherId].left = true;
195         console::log("philosopher",philosopherId );
196         console::log("picks fork", philosopherId);
197     }
198 }
199 else{
200     console::log("philosopher",philosopherId );
201     console::log("waiting for fork", philosopherId);
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }

/** Driver code ***/
int main(){
    DiningPhilosophers diningPhilosophers = DiningPhilosophers(5);
    diningPhilosophers.startDinner();
}

(END)
```

## Output

## Question

Dining Philosophers Algorithm implemented using Semaphores

## Code

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
```

```
#include <semaphore.h>

typedef struct {
    int position;
    int count;
    sem_t *forks;
    sem_t *lock;
} params_t;

void initialize_semaphores(sem_t *lock, sem_t *forks, int num_forks);
void run_all_threads(pthread_t *threads, sem_t *forks, sem_t *lock, int num_philosophers);

void *philosopher(void *params);
void think(int position);
void eat(int position);

int main(int argc, char *args[])
{
    int num_philosophers = 5;

    sem_t lock;
    sem_t forks[num_philosophers];
    pthread_t philosophers[num_philosophers];

    initialize_semaphores(&lock, forks, num_philosophers);
    run_all_threads(philosophers, forks, &lock, num_philosophers);
    pthread_exit(NULL);
}

void initialize_semaphores(sem_t *lock, sem_t *forks, int num_forks)
{
    int i;
    for(i = 0; i < num_forks; i++) {
        sem_init(&forks[i], 0, 1);
    }

    sem_init(lock, 0, num_forks - 1);
}

void run_all_threads(pthread_t *threads, sem_t *forks, sem_t *lock, int num_philosophers)
{
    int i;
    for(i = 0; i < num_philosophers; i++) {
        params_t *arg = malloc(sizeof(params_t));
        arg->position = i;
        arg->count = num_philosophers;
        arg->lock = lock;
        arg->forks = forks;

        pthread_create(&threads[i], NULL, philosopher, (void *)arg);
    }
}
```

```
void *philosopher(void *params)
{
    int i;
    params_t self = *(params_t *)params;

    for(i = 0; i < 3; i++) {
        think(self.position);

        sem_wait(self.lock);
        sem_wait(&self.forks[self.position]);
        sem_wait(&self.forks[(self.position + 1) % self.count]);
        eat(self.position);
        sem_post(&self.forks[self.position]);
        sem_post(&self.forks[(self.position + 1) % self.count]);
        sem_post(self.lock);
    }

    think(self.position);
    pthread_exit(NULL);
}

void think(int position)
{
    printf("Philosopher %d thinking...\n", position);
}

void eat(int position)
{
    printf("Philosopher %d eating...\n", position);
}
```

## Code Preview

```

yash@hephaestus: ~
bat DiningPhilosopherSemaphore.cpp

File: DiningPhilosopherSemaphore.cpp

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <semaphore.h>
5
6 typedef struct {
7     int position;
8     int count;
9     sem_t *forks;
10    sem_t *lock;
11 } params_t;
12
13 void initialize_semaphores(sem_t *lock, sem_t *forks, int num_forks);
14 void run_all_threads(pthread_t *threads, sem_t *forks, sem_t *lock, int num_philosophers);
15
16 void *philosopher(void *params);
17 void think(int position);
18 void eat(int position);
19
20 int main(int argc, char *args[])
21 {
22     int num_philosophers = 5;
23
24     sem_t lock;
25     sem_t forks[num_philosophers];
26     pthread_t philosophers[num_philosophers];
27
28     initialize_semaphores(&lock, forks, num_philosophers);
29     run_all_threads(philosophers, forks, &lock, num_philosophers);
30     pthread_exit(NULL);
31 }
32
33 void initialize_semaphores(sem_t *lock, sem_t *forks, int num_forks)
34 {
35     int i;
36     for(i = 0; i < num_forks; i++) {
37         sem_init(&forks[i], 0, 1);
38     }
39
40     sem_init(lock, 0, num_forks - 1);
41 }
42
43 void run_all_threads(pthread_t *threads, sem_t *forks, sem_t *lock, int num_philosophers)
44 {
45     int i;
46     for(i = 0; i < num_philosophers; i++) {
47         params_t *arg = malloc(sizeof(params_t));
48         arg->position = i;
49         arg->count = num_philosophers;
50         arg->lock = lock;
51         arg->forks = forks;
52
53         pthread_create(&threads[i], NULL, philosopher, (void *)arg);
54     }
55 }
56
57 void *philosopher(void *params)
58 {
59     int i;
60     params_t self = *(params_t *)params;
61
62     for(i = 0; i < 3; i++) {
63         think(self.position);
64
65         sem_wait(self.lock);
66         sem_wait(&self.forks[self.position]);
67         sem_wait(&self.forks[(self.position + 1) % self.count]);
68         eat(self.position);
69         sem_post(&self.forks[self.position]);
70     }
71 }
72
73 MYS

```

```

yash@hephaestus: ~
bat DiningPhilosopherSemaphore.cpp

32 void initialize_semaphores(sem_t *lock, sem_t *forks, int num_forks)
33 {
34     int i;
35     for(i = 0; i < num_forks; i++) {
36         sem_init(&forks[i], 0, 1);
37     }
38
39     sem_init(lock, 0, num_forks - 1);
40 }
41
42 void run_all_threads(pthread_t *threads, sem_t *forks, sem_t *lock, int num_philosophers)
43 {
44     int i;
45     for(i = 0; i < num_philosophers; i++) {
46         params_t *arg = malloc(sizeof(params_t));
47         arg->position = i;
48         arg->count = num_philosophers;
49         arg->lock = lock;
50         arg->forks = forks;
51
52         pthread_create(&threads[i], NULL, philosopher, (void *)arg);
53     }
54 }
55
56 void *philosopher(void *params)
57 {
58     int i;
59     params_t self = *(params_t *)params;
60
61     for(i = 0; i < 3; i++) {
62         think(self.position);
63
64         sem_wait(self.lock);
65         sem_wait(&self.forks[self.position]);
66         sem_wait(&self.forks[(self.position + 1) % self.count]);
67         eat(self.position);
68         sem_post(&self.forks[self.position]);
69     }
70 }
71
72 MYS

```

yash@hephaestus: ~

bat DiningPhilosopherSemaphore.cpp

```
50     arg->lock = lock;
51     arg->forks = forks;
52
53     if(pthread_create(&threads[i], NULL, philosopher, (void *)arg));
54     }
55
56     // Diagnostics
57 void *philosopher(void *params)
58 {
59     int i;
60     params_t self = *(params_t *)params;
61
62     for(i = 0; i < 3; i++) {
63         think(self.position);
64
65         sem_wait(self.lock);
66         sem_wait(&self.forks[self.position]);
67         sem_wait(&self.forks[(self.position + 1) % self.count]);
68         eat(self.position);
69         sem_post(&self.forks[self.position]);
70         sem_post(&self.forks[(self.position + 1) % self.count]);
71         sem_post(self.lock);
72     }
73
74     // Print final position
75     think(self.position);
76     pthread_exit(NULL);
77 } // DiningPhilosopherSemaphore.cpp
78 void think(int position) Semaphore.cpp ##### Output
79 {
80     printf("Philosopher %d thinking...\n", position);
81 }
82
83 void eat(int position)
84 {
85     printf("Philosopher %d eating...\n", position);
86 }
```

## Output

## Question

## Implementation of Bankers Algorithm

## Code

```
/**  
 * @author YashKumarVerma / 19BCE2669  
 * @for: Lab Digital Assignment 3  
 * @description: Implementation of Bankers Algorithm  
 */  
  
#include <bits/stdc++.h>  
  
using namespace std;  
  
/** Class to handle all operating and keep data **/  
class BankerAlgorithm {  
public:  
    int p;  
    bool come;  
    int keepOn;  
    int Max[10][3];  
    int current[3];  
    int need[10][3];  
    bool executed[10];  
    int available[10][3];  
    int allocation[10][3];  
  
    BankerAlgorithm() { int keepOn = 1; }  
  
    void TakeInput() {  
        cout << "Enter No. of processes: ";  
        cin >> p;  
        cout << "\n";  
        cout << "Enter the current resources: ";  
        cin >> current[0] >> current[1] >> current[2];  
  
        for (int i = 0; i < p; ++i) {  
            cout << "[19BCE2669] \t\t > Process P" << i + 1 << " Details" <<  
endl;  
            cout << "Enter Allocation : ";  
            cin >> allocation[i][0] >> allocation[i][1] >> allocation[i][2];  
            cout << "Enter Max :";  
            cin >> Max[i][0] >> Max[i][1] >> Max[i][2];  
            need[i][0] = Max[i][0] - allocation[i][0];  
            need[i][1] = Max[i][1] - allocation[i][1];  
            need[i][2] = Max[i][2] - allocation[i][2];  
        }  
  
        cout << "[19BCE2669] \t\t > Table for Bankers Algo" << endl << endl;  
        cout << "Initial Resources: " << current[0] << " " << current[1] << " "  
        << current[2] << endl  
        << endl;  
        cout << "Process      Max      Allocation      Need" << endl;  
  
        for (int i = 0; i < p; ++i) {  
            cout << "  P" << i + 1 << "      "  
            << "  " << Max[i][0] << "  " << Max[i][1] << "  " << Max[i][2]  
        }  
    }  
};
```

```

        << "      "
        << " " << allocation[i][0] << " " << allocation[i][1] << " "
        << allocation[i][2] << "      "
        << " " << need[i][0] << " " << need[i][1] << " " << need[i][2]
        << endl;
    }
}

void ProcessData() {
    Calculate();
    while (keepOn) {
        int val, pro;
        cout << "\n\nSelect Below operations:" << endl << endl;
        cout << "1.Change Max of process: " << endl;
        cout << "2.Change Allocation of process" << endl;
        cout << "3.Change Initial Resources" << endl;
        cout << "4.Exit" << endl;
        cin >> val;

        if (val == 1) {
            cout << "\n\nEnter Process No: ";
            cin >> pro;
            cout << "\nEnter New Max: ";
            cin >> Max[pro - 1][0] >> Max[pro - 1][1] >> Max[pro - 1][2];
        } else if (val == 2) {
            cout << "\n\nEnter Process No: ";
            cin >> pro;
            cout << "\nEnter New Allocation: ";
            cin >> allocation[pro - 1][0] >> allocation[pro - 1][1] >>
                allocation[pro - 1][2];
        } else if (val == 3) {
            cout << "\nEnter Initial Resources: ";
            cin >> current[0] >> current[1] >> current[2];
        } else {
            break;
        }
    }
    Calculate();
}

void Calculate() {
    IMP();
    int i, j;
    for (i = 0; i < p; ++i) {
        for (j = 0; j < p; ++j) {
            while (executed[j] && j < p - 1) {
                j++;
            }
            if (need[j][0] <= current[0] && need[j][1] <= current[1] &&
                need[j][2] <= current[2]) {
                if (!executed[j]) {
                    executed[j] = true;
                    current[0] += allocation[j][0];
                    current[1] += allocation[j][1];
                }
            }
        }
    }
}

```

```
        current[2] += allocation[j][2];
        cout << "\nProcess P" << j + 1;
        cout << "\nCurrent: " << current[0] << " " << current[1] << " "
            << current[2] << "\n";
        cout << "\nProcess executed without deadlock";
        come = true;
        break;
    }
}
if (!come) {
    cout << "\n\t\t\tDead lock\n\n";
    break;
} else {
    come = false;
}
}
}

void IMP() {
    come = false;
    for (int i = 0; i < 10; ++i) {
        executed[i] = false;
    }
}
};

int main() {
    BankerAlgorithm handler = BankerAlgorithm();
    handler.TakeInput();
    handler.ProcessData();
    return 0;
}
```

## Code Preview

```
yash@hephaestus:~  
bat BankersAlgorithm.cpp  
  
34 cout << "[19BCE2669] \t\t> Process P" << i + 1 << " Details" << endl;  
35 cout << "Enter Allocation : ";  
36 cin >> allocation[i][0] >> allocation[i][1] >> allocation[i][2];  
37 cout << "Enter Max :";  
38 cin >> Max[i][0] >> Max[i][1] >> Max[i][2];  
39 need[i][0] = Max[i][0] - allocation[i][0];  
40 need[i][1] = Max[i][1] - allocation[i][1];  
41 need[i][2] = Max[i][2] - allocation[i][2];  
42 }  
43 cout << "[19BCE2669] \t\t> Table for Bankers Algo" << endl << endl;  
44 cout << "Initial Resources: " << current[0] << " " << current[1] << " "  
45 << current[2] << endl;  
46 << endl;  
47 cout << "Process      Max      Allocation      Need" << endl;  
48 handler.TakeInput();  
49  
50 for (int i = 0; i < p; i++) {  
51     cout << " P" << i + 1 << "      " ;  
52     cout << " " << Max[i][0] << " " << Max[i][1] << " " << Max[i][2]  
53     cout << "      "  
54     cout << " " << allocation[i][0] << " " << allocation[i][1] << " "  
55     cout << " " << allocation[i][2] << " "  
56     cout << " " << need[i][0] << " " << need[i][1] << " " << need[i][2]  
57     cout << endl;  
58 }  
59 // DinningPhilosopherMain.cpp  
60 // DinningPhilosopherSemaphore.cpp  
61 void ProcessData() {  
62     Calculate();  
63     while (keepOn) {  
64         int val, pro;  
65         cout << "\n\nSelect Below operations:" << endl << endl;  
66         cout << "1.Change Max of process: " << endl;  
67         cout << "2.Change Allocation of process" << endl;  
68         cout << "3.Change Initial Resources" << endl;  
69         cout << "4.Exit" << endl;  
70         cin >> val;  
71     }  
72 }
```

```

yash@hephaestus: ~
bat BankersAlgorithm.cpp

60 void ProcessData() {
61     Calculate();
62     while (keepOn) {
63         int val, pro;
64         cout << "\n\nSelect Below operations:" << endl << endl;
65         cout << "1.Change Max of process: " << endl;
66         cout << "2.Change Allocation of process" << endl;
67         cout << "3.Change Initial Resources" << endl;
68         cout << "4.Exit" << endl;
69         cin >> val;
70
71         if (val == 1) {
72             cout << "\nEnter Process No: ";
73             cin >> pro;
74             cout << "\nEnter New Max: ";
75             cin >> Max[pro - 1][0] >> Max[pro - 1][1] >> Max[pro - 1][2];
76         } else if (val == 2) {
77             cout << "\nEnter Process No: ";
78             cin >> pro;
79             cout << "\nEnter New Allocation: ";
80             cin >> allocation[pro - 1][0] >> allocation[pro - 1][1] >>
81             allocation[pro - 1][2];
82         } else if (val == 3) {
83             cout << "\nEnter Initial Resources: " ##### Code Preview
84             cin >> current[0] >> current[1] >> current[2]; ([https://i.imgur.com/9zhLoF0.png](https://i.imgur.com/9zhLoF0.png)
85         } else {
86             break;
87         }
88     }
89     Calculate();
90 }
91
92
93 void Calculate() {
94     IMP();
95     int i, j;
96     for (i = 0; i < p; ++i) {
97         for (j = 0; j < p; ++j) {
98             while (executed[j] && j < p - 1) {
99                 j++;
100             }
101             if (need[j][0] <= current[0] && need[j][1] <= current[1] &&
102                 need[j][2] <= current[2]) {
103                 if (!executed[j]) {
104                     executed[j] = true;
105                     current[0] += allocation[j][0];
106                     current[1] += allocation[j][1];
107                     current[2] += allocation[j][2];
108                     cout << "\nProcess P" << j + 1;
109                     cout << "\nCurrent: " << current[0] << " " << current[1] << " " << current[2] << "\n";
110                     cout << "\nProcess executed without deadlock"; handler.ProcessData();
111                     come = true;
112                     break;
113                 }
114             }
115         }
116     }
117     if (!come) {
118         cout << "\n\t\t\tDead lock\n\n";
119         break;
120     } else {
121         come = false; semaphore.cpp
122     }
123 }
124
125
126 void IMP() {
127     come = false;
128     for (int i = 0; i < 10; ++i) {
129         executed[i] = false;
130     }
131 }
132 }

133 int main() {
134     BankerAlgorithm handler = BankerAlgorithm();
135     handler.TakeInput();
136     handler.ProcessData(); ##### Output
137
138     return 0;
139 }

(END)

```

## Output

```

yash@hephaestus: ~
~/BankersAlgorithm
→ assignment3 git:(19BCE2669) ✘ g++ BankersAlgorithm.cpp -o BankersAlgorithm
→ assignment3 git:(19BCE2669) ✘ ./BankersAlgorithm
Enter No. of processes: 3
Enter the current resources: 5 5 5
[19BCE2669] > Process P1 Details
Enter Allocation : 3 4 5
Enter Max :5 6 7
[19BCE2669] > Process P2 Details
Enter Allocation : 2 0 1
Enter Max :8 4 7
[19BCE2669] > Process P3 Details
Enter Allocation : 1 1 1
Enter Max :6 7 8
[19BCE2669] javaCodes > Table for Bankers Algo
Initial Resources: 5 5 5
Process   Max     Allocation    Need
P1      5 6 7  assigned 3 4 5    2 2 2
P2      8 4 7  allocated 2 0 1    6 4 6
P3      6 7 8  allocated 1 1 1    5 6 7
Process P1
Current: 8 9 10
Process executed without deadlock
Process P2
Current: 10 9 11
Process executed without deadlock
Process P3
Current: 11 10 12
Process executed without deadlock
Select Below operations:
1.Change Max of process
2.Change Allocation of process

```

## Question

Implementing Producers and Consumers Problem using Semaphore

### Code

```

/**
 * @author YashKumarVerma / 19BCE2669
 * @for: Lab Digital Assignment 3
 * @description: Implementation of producer consumer problem using
semaphores
 */

/** defining constants for application ***/
#define BUFFER_SIZE 10

/** Including Required headers */
#include <iostream>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

using namespace std;

/**
 * @class Config
 * @description To store code configurations and local state
 */
class Config {

```

```

public:
    static int MaxItems;
    static int BufferSize;
    static sem_t empty;
    static sem_t full;
    static int in;
    static int out;
    static int buffer[BUFFER_SIZE];
    static pthread_mutex_t mutex;
};

/** Initializing All App state **/
int Config::MaxItems = 10;
int Config::BufferSize = BUFFER_SIZE;
sem_t Config::empty;
sem_t Config::full;
int Config::in = 0;
int Config::out = 0;
int Config::buffer[BUFFER_SIZE];
pthread_mutex_t Config::mutex;

/**
 * @class Semaphore
 * @description Class to contain definitions for Producer and Consumer
 */
class Semaphore {
public:
    /* consumer process */
    static void* Consumer(void* cno) {
        for (int i = 0; i < Config::MaxItems; i++) {
            sem_wait(&Config::full);
            pthread_mutex_lock(&Config::mutex);
            int item = Config::buffer[Config::out];
            printf("Consumer %d: Remove Item %d from %d\n", *((int*)cno),
item, Config::out);
            Config::out = (Config::out + 1) % Config::BufferSize;
            pthread_mutex_unlock(&Config::mutex);
            sem_post(&Config::empty);
        }
    }

    /* producer process */
    static void* Producer(void* pno) {
        int item;
        for (int i = 0; i < Config::MaxItems; i++) {
            item = rand(); // Produce an random item
            sem_wait(&Config::empty);
            pthread_mutex_lock(&Config::mutex);
            Config::buffer[Config::in] = item;
            printf("Producer %d: Insert Item %d at %d\n", *((int*)pno),
Config::buffer[Config::in], Config::in);
            Config::in = (Config::in + 1) % Config::BufferSize;
            pthread_mutex_unlock(&Config::mutex);
            sem_post(&Config::full);
        }
    }
};

```

```
        }
    }
};

/* main caller function */
int main() {

    pthread_t pro[5], con[5];
    pthread_mutex_init(&Config::mutex, NULL);
    sem_init(&Config::empty, 0, Config::BufferSize);
    sem_init(&Config::full, 0, 0);

    /* Just used for numbering the producer and consumer */
    int a[5] = { 1, 2, 3, 4, 5 };
    for (int i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, Semaphore::Producer, (void*)&a[i]);
    }
    for (int i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, Semaphore::Consumer, (void*)&a[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for (int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }

    pthread_mutex_destroy(&Config::mutex);
    sem_destroy(&Config::empty);
    sem_destroy(&Config::full);

    return 0;
}
```

## Code Preview

yash@hephaestus: ~

bat ProducerAndConsumerSemaphore.cpp

```

File: ProducerAndConsumerSemaphore.cpp

1  /* Code
2  * @author YashKumarVerma / 19BCE2669
3  * @for: Lab Digital Assignment 3
4  * @description: Implementation of producer consumer problem using semaphores
5  */
6
7  /** defining constants for application **/
8  #define BUFFER_SIZE 10
9
10 /* Including Required headers */
11 #include <iostream>
12 #include <thread.h>
13 #include <semaphore.h>
14 #include <stdlib.h>
15 #include <stdio.h>
16
17 using namespace std;
18
19 /* 19BCE2669-DA3.md
20 * @class Config
21 * @description To store code configurations and local state
22 */
23 class Config {
24 public:
25     static int MaxItems;
26     static int BufferSize;
27     static sem_t empty;
28     static sem_t full;
29     static int in;
30     static int out;
31     static int buffer[BUFFER_SIZE];
32     static pthread_mutex_t mutex;
33 };
34
35 /* Initializing All App state */

```

yash@hephaestus: ~

bat ProducerAndConsumerSemaphore.cpp

```

34
35 /* Initializing All App state */
36 int Config::MaxItems = 10;
37 int Config::BufferSize = BUFFER_SIZE;
38 sem_t Config::empty;
39 sem_t Config::full;
40 int Config::in = 0;
41 int Config::out = 0;
42 int Config::buffer[BUFFER_SIZE];
43 pthread_mutex_t Config::mutex;
44
45 /* assignments
46
47 * @class Semaphore
48 * @description Class to contain definitions for Producer and Consumer
49 */
50 class Semaphore {
51 public:
52     /* consumer process */
53     static void* Consumer(void* cno) {
54         for (int i = 0; i < Config::MaxItems; i++) {
55             sem_wait(&Config::full);
56             pthread_mutex_lock(&Config::mutex);
57             int item = Config::buffer[Config::out];
58             printf("Consumer %d: Remove Item %d from %d\n", *((int*)cno), item, Config::out);
59             Config::out = (Config::out + 1) % Config::BufferSize;
60             pthread_mutex_unlock(&Config::mutex);
61             sem_post(&Config::empty);
62         }
63     }
64
65     /* producer process */
66     static void* Producer(void* pno) {
67         int item;
68         for (int i = 0; i < Config::MaxItems; i++) {
69             item = rand(); // Produce an random item
70             sem_wait(&Config::empty);
71             pthread_mutex_lock(&Config::mutex);
72             Config::buffer[Config::in] = item;
73         }
74     }

```

```

yash@hephaestus: ~
bat ProducerAndConsumerSemaphore.cpp

68     item = rand(); // Produce an random item
69     sem_wait(&Config::empty);
70     pthread_mutex_lock(&Config::mutex);           ##### Code Preview
71     Config::buffer[Config::in] = item;
72     printf("Producer %d: Insert Item %d at %d\n", *(int*)pno, Config::buffer[Config::in], Config::in);
73     Config::in = (Config::in + 1) % Config::BufferSize;
74     pthread_mutex_unlock(&Config::mutex);
75     sem_post(&Config::full);
76 }
77 }
78 assignments
79
80 /* main caller function */
81 int main(){
82
83     pthread_t pro[5], con[5];
84     pthread_mutex_init(&Config::mutex, NULL);          ##### Output
85     sem_init(&Config::empty, 0, Config::BufferSize);
86     sem_init(&Config::full, 0, 0);
87
88     /* Just used for numbering the producer and consumer */
89     int a[5] = { 1, 2, 3, 4, 5 };
90     for (int i = 0; i < 5; i++) {
91         pthread_create(&pro[i], NULL, Semaphore::Producer, (void*)&a[i]);
92     }
93     for (int i = 0; i < 5; i++) {
94         pthread_create(&con[i], NULL, Semaphore::Consumer, (void*)&a[i]);
95     }
96
97     for (int i = 0; i < 5; i++) {           ##### Code Preview
98         pthread_join(pro[i], NULL);
99     }
100    for (int i = 0; i < 5; i++) {
101        pthread_join(con[i], NULL);
102    }
103
104    pthread_mutex_destroy(&Config::mutex);
105    sem_destroy(&Config::empty);
106
107
108    return 0;  //main()
109 }

78 };
109
110 logposts
111
112 /* main caller function */
113 int main(){
114     college
115
116     pthread_t pro[5], con[5];
117     pthread_mutex_init(&Config::mutex, NULL);          ##### Code Preview
118     sem_init(&Config::empty, 0, Config::BufferSize);
119     sem_init(&Config::full, 0, 0);
120
121     /* Just used for numbering the producer and consumer */
122     int a[5] = { 1, 2, 3, 4, 5 };
123     for (int i = 0; i < 5; i++) {
124         pthread_create(&pro[i], NULL, Semaphore::Producer, (void*)&a[i]);           ##### Output
125     }
126     for (int i = 0; i < 5; i++) {
127         pthread_create(&con[i], NULL, Semaphore::Consumer, (void*)&a[i]);
128     }
129
130     for (int i = 0; i < 5; i++) {           ##### Code Preview
131         pthread_join(pro[i], NULL);
132     }
133     for (int i = 0; i < 5; i++) {
134         pthread_join(con[i], NULL);
135     }
136
137     pthread_mutex_destroy(&Config::mutex);
138     sem_destroy(&Config::empty);
139     sem_destroy(&Config::full);
140
141
142     return 0;  //main()
143 }

(END)

```

## Output

```
yash@hephaestus: ~
yash@hephaestus: ~/Desktop/files/works/foam-notes/college/assignments/os/assignment3

→ assignment3 git:(19BCE2669) ✘ ./ProducerAndConsumer
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 1714636915 at 1
Producer 1: Insert Item 1957747793 at 2
Producer 2: Insert Item 719885386 at 3
Producer 2: Insert Item 596516649 at 4
Producer 2: Insert Item 1189641421 at 5
Producer 4: Insert Item 846930886 at 6
Producer 3: Insert Item 1681692777 at 7
Producer 5: Insert Item 424238335 at 8
Producer 1: Insert Item 1649760492 at 9
Consumer 2: Remove Item 1804289383 from 0
Consumer 3: Remove Item 1714636915 from 1
Producer 2: Insert Item 1025202362 at 0
Producer 4: Insert Item 1350490027 at 1
Consumer 5: Remove Item 1957747793 from 2
Consumer 5: Remove Item 719885386 from 3
Consumer 5: Remove Item 596516649 from 4
Producer 5: Insert Item 1102520059 at 2
Producer 5: Insert Item 1540383426 at 3
Producer 3: Insert Item 783368690 at 4
Consumer 3: Remove Item 1189641421 from 5
Consumer 2: Remove Item 846930886 from 6
Consumer 5: Remove Item 1681692777 from 7
Consumer 4: Remove Item 424238335 from 8
Consumer 3: Remove Item 1649760492 from 9
Producer 2: Insert Item 1967513926 at 5
Producer 3: Insert Item 1303455736 at 6
Producer 4: Insert Item 1365180540 at 7
Consumer 5: Remove Item 1025202362 from 0
Producer 1: Insert Item 2044897763 at 8
Consumer 4: Remove Item 1350490027 from 1
Producer 5: Insert Item 304089172 at 9
Consumer 3: Remove Item 1102520059 from 2
Consumer 2: Remove Item 1540383426 from 3
Consumer 5: Remove Item 783368690 from 4
Producer 2: Insert Item 35005211 at 0
Consumer 4: Remove Item 1967513926 from 5
Producer 3: Insert Item 521595268 at 1
Producer 4: Insert Item 1369133069 at 2
Producer 5: Insert Item 2145174067 from 8
Producer 5: Insert Item 1125898167 at 6
Consumer 5: Remove Item 468703135 from 9
Consumer 3: Remove Item 1101513929 from 0
Consumer 3: Remove Item 1801979802 from 1
Producer 3: Insert Item 2089018456 at 7
Consumer 4: Remove Item 1315634022 from 2
Consumer 4: Remove Item 1059961393 from 3
Consumer 4: Remove Item 635723058 from 4
Consumer 4: Remove Item 1369133069 from 5
Consumer 2: Remove Item 1125898167 from 6
Consumer 3: Remove Item 2089018456 from 7
Producer 5: Insert Item 1653377373 at 8
Producer 2: Insert Item 628175011 at 9
Producer 3: Insert Item 859484421 at 0
Producer 3: Insert Item 608413784 at 1
Producer 3: Insert Item 756898537 at 2
Producer 5: Insert Item 1914544919 at 3
Producer 5: Insert Item 1734575198 at 4
Producer 1: Insert Item 1656478042 at 5
Producer 1: Insert Item 1973594324 at 6
Consumer 2: Remove Item 1653377373 from 0
Consumer 2: Remove Item 628175011 from 9
Producer 4: Insert Item 1131176229 at 7
Producer 4: Insert Item 149798315 at 8
Producer 4: Insert Item 2038664370 at 9
Consumer 1: Remove Item 859484421 from 0
Consumer 1: Remove Item 608413784 from 1
Consumer 1: Remove Item 756898537 from 2
Consumer 1: Remove Item 1914544919 from 3
Consumer 1: Remove Item 1734575198 from 4
Consumer 1: Remove Item 1656478042 from 5
Consumer 1: Remove Item 1973594324 from 6
Consumer 1: Remove Item 1131176229 from 7
Consumer 1: Remove Item 149798315 from 8
Consumer 1: Remove Item 2038664370 from 9
→ assignment3 git:(19BCE2669) ✘
```

## Question

Implement Readers and Writers Problem using Semaphores

### Code

```
/*
 * @author YashKumarVerma / 19BCE2669
 * @for: Lab Digital Assignment 3
```

```
* @description: Implementation of readers-writers problem using semaphores
*/



#include<iostream>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

using namespace std;

sem_t mutex;
sem_t wrt;
int data = 0;
int readCount = 0;

/***
 * @class Semaphore
 * @description: to define reader and writer behavior
 */
class Semaphore{
public:

    static void *Reader(void *arg){
        int f = ((intptr_t)arg);

        sem_wait(&mutex);
        readCount += 1;
        if(readCount == 1){
            sem_wait(&wrt);
        }

        sem_post(&mutex);
        std::cout << "Data read by the reader : " << f << " is " << data <<
std::endl;
        sleep(1);

        sem_wait(&mutex);
        readCount -= 1;
        if(readCount == 0){
            sem_post(&wrt);
        }
        sem_post(&mutex);
    }

    static void *Writer(void *arg){
        int f = ((intptr_t)arg);
        sem_wait(&wrt);
        data++;
        std::cout << "Data written by the writer : " << f << " is " << data
<< std::endl;
        sleep(1);
        sem_post(&wrt);
    }
}
```

```
    }

};

/** Main caller function */
int main() {

    int i, b;
    pthread_t readerThreadId[8], writerThreadId[8];
    sem_init(&mutex, 0, 1);
    sem_init(&wrt, 0, 1);
    for (i=0;i<=5;i++) {
        pthread_create(&writerThreadId[i], NULL, Semaphore::Writer, (void *)(intptr_t)i);
        pthread_create(&readerThreadId[i], NULL, Semaphore::Reader, (void *)(intptr_t)i);
    }
    for (i=0;i<=5;i++) {
        pthread_join(writerThreadId[i], NULL);
        pthread_join(readerThreadId[i], NULL);
    }
    return 0;
}
```

## Code Preview

```

yash@hephaestus: ~
bat ReaderAndWriterSemaphore.cpp

File: ReaderAndWriterSemaphore.cpp

1  /**
2  * @author YashKumarVerma / 19BCE2669
3  * @for: Lab Digital Assignment 3
4  * @description: Implementation of readers-writers problem using semaphores
5  */
6
7 #include<iostream>
8 #include <stdio.h>
9 #include <thread.h>
10 #include <semaphore.h>
11 #include <unistd.h>
12
13 using namespace std;
14
15 sem_t mutex;
16 sem_t wrt;
17 int data = 0;
18 int readCount = 0;
19
20 /**
21 * @class Semaphore
22 * @description: to define reader and writer behavior
23 */
24 class Semaphore{
25 public:
26     // Constructor
27     Semaphore();
28     static void *Reader(void *arg);
29     static void *Writer(void *arg);
30
31     // Destructor
32     ~Semaphore();
33
34     // Methods
35     void read();
36     void write();
37
38     // Private members
39     sem_t mutex;
40     sem_t wrt;
41     int data;
42     int readCount;
43
44     // Private methods
45     void sem_post(sem_t *sem);
46     void sem_wait(sem_t *sem);
47
48     // Private variables
49     static void *Reader(void *arg);
50     static void *Writer(void *arg);
51
52     // Private functions
53     void assignment();
54     void dbm();
55
56     // Private variables
57     static void *Reader(void *arg);
58     static void *Writer(void *arg);
59
60     // Main caller function
61     int main();
62
63     // Thread creation
64     pthread_t readerThreadId[8], writerThreadId[8];
65     sem_init(&mutex, 0, 1);
66     sem_init(&wrt, 0, 1);
67     for (i=0;i<5;i++){
68         pthread_create(&writerThreadId[i], NULL, Semaphore::Writer, (void *)(intptr_t)i);
69         pthread_create(&readerThreadId[i], NULL, Semaphore::Reader, (void *)(intptr_t)i);
70     }
71     for (i=0;i<5;i++){
72         pthread_join(writerThreadId[i], NULL);
73         pthread_join(readerThreadId[i], NULL);
74     }
75 }

END)

```

File: ReaderAndWriterSemaphore.cpp

```

yash@hephaestus: ~
bat ReaderAndWriterSemaphore.cpp

File: ReaderAndWriterSemaphore.cpp

34     sem_wait(&wrt);
35 }
36
37 // Reader function
38 void Reader(void *arg){
39     int f = ((intptr_t)arg);
40     sem_wait(&mutex);
41     readCount += 1;
42     if(readCount == 0){
43         sem_post(&wrt);
44     }
45     std::cout << "Data read by the reader : " << f << " is " << data << std::endl;
46     sleep(1);
47     sem_post(&mutex);
48     readCount -= 1;
49     if(readCount == 0){
50         sem_post(&wrt);
51     }
52     std::cout << "Data written by the writer : " << f << " is " << data << std::endl;
53     sleep(1);
54     sem_post(&wrt);
55 }
56
57 // Main caller function
58 int main(){
59     // Constructor
60     Semaphore();
61     // Thread creation
62     pthread_t readerThreadId[8], writerThreadId[8];
63     sem_init(&mutex, 0, 1);
64     sem_init(&wrt, 0, 1);
65     for (i=0;i<5;i++){
66         pthread_create(&writerThreadId[i], NULL, Semaphore::Writer, (void *)(intptr_t)i);
67         pthread_create(&readerThreadId[i], NULL, Semaphore::Reader, (void *)(intptr_t)i);
68     }
69     for (i=0;i<5;i++){
70         pthread_join(writerThreadId[i], NULL);
71         pthread_join(readerThreadId[i], NULL);
72     }
73     return 0;
74 }

END)

```

File: ReaderAndWriterSemaphore.cpp

```

yash@hephaestus: ~
bat ReaderAndWriterSemaphore.cpp

File: ReaderAndWriterSemaphore.cpp

34     sem_wait(&wrt);
35 }
36
37 // Reader function
38 void Reader(void *arg){
39     int f = ((intptr_t)arg);
40     sem_wait(&mutex);
41     readCount += 1;
42     if(readCount == 0){
43         sem_post(&wrt);
44     }
45     std::cout << "Data read by the reader : " << f << " is " << data << std::endl;
46     sleep(1);
47     sem_post(&mutex);
48     readCount -= 1;
49     if(readCount == 0){
50         sem_post(&wrt);
51     }
52     std::cout << "Data written by the writer : " << f << " is " << data << std::endl;
53     sleep(1);
54     sem_post(&wrt);
55 }
56
57 // Main caller function
58 int main(){
59     // Constructor
60     Semaphore();
61     // Thread creation
62     pthread_t readerThreadId[8], writerThreadId[8];
63     sem_init(&mutex, 0, 1);
64     sem_init(&wrt, 0, 1);
65     for (i=0;i<5;i++){
66         pthread_create(&writerThreadId[i], NULL, Semaphore::Writer, (void *)(intptr_t)i);
67         pthread_create(&readerThreadId[i], NULL, Semaphore::Reader, (void *)(intptr_t)i);
68     }
69     for (i=0;i<5;i++){
70         pthread_join(writerThreadId[i], NULL);
71         pthread_join(readerThreadId[i], NULL);
72     }
73     return 0;
74 }

END)

```

## Output

## Time dependent Execution

```
yash@hephaestus: ~
./ReaderAndWriter
Data written by the writer : 0 is 1
Data written by the writer : 1 is 2
Data read by the reader : 1 is 2
Data read by the reader : 0 is 2
Data read by the reader : 3 is 2
Data read by the reader : 2 is 2
Data read by the reader : 4 is 2
Data read by the reader : 5 is 2
Data written by the writer : 2 is 3
[https://i.imgur.com/vcnPfhp.png](https://i.imgur.com/vcnPfhp.png)
[https://i.imgur.com/vcnPfhp.png](https://i.imgur.com/vcnPfhp.png)

## Question
Implement Readers and Writers Problem using Semaphores

> #### Code

[https://i.imgur.com/l4zwg0l.png](https://i.imgur.com/l4zwg0l.png)
[https://i.imgur.com/0h3Y9Ml.png](https://i.imgur.com/0h3Y9Ml.png)
[https://i.imgur.com/46C04hs.png](https://i.imgur.com/46C04hs.png)

#### Output
[https://i.imgur.com/0h3Y9Ml.png](https://i.imgur.com/0h3Y9Ml.png)
[https://i.imgur.com/46C04hs.png](https://i.imgur.com/46C04hs.png)

---
```

File Explorer:

- assignments
  - dbms
  - java
  - javaCodes
- os
  - assignment2
  - assignment3
    - 19BCE2669-DA3.md
    - a.out
    - BankersAlgorithm
      - BankersAlgorithm.cpp
      - DiningPhilosopher/Monitors.cpp
      - DiningPhilosopher/Semaphore.cpp
    - ProducerAndConsumer
      - ProducerAndConsumerSemaphore.cpp
    - ReaderAndWriter
      - ReaderAndWriterSemaphore.cpp
  - assignment2.md
- MySQL

```
yash@hephaestus: ~
yash@hephaestus: ~/Desktop/files/works/foam-notes/college/assignments/os/assignment3
./ReaderAndWriter
Data written by the writer : 0 is 1
Data written by the writer : 1 is 2
Data read by the reader : 1 is 2
Data read by the reader : 0 is 2
Data read by the reader : 3 is 2
Data read by the reader : 2 is 2
Data read by the reader : 4 is 2
Data read by the reader : 5 is 2
Data written by the writer : 2 is 3
Data written by the writer : 3 is 4
Data written by the writer : 4 is 5
Data written by the writer : 5 is 6
Execution time: 0h:00m:07s sec
[https://i.imgur.com/vcnPfhp.png](https://i.imgur.com/vcnPfhp.png)
[https://i.imgur.com/vcnPfhp.png](https://i.imgur.com/vcnPfhp.png)

## Question
Implement Readers and Writers Problem using Semaphores

> #### Code

[https://i.imgur.com/l4zwg0l.png](https://i.imgur.com/l4zwg0l.png)
[https://i.imgur.com/0h3Y9Ml.png](https://i.imgur.com/0h3Y9Ml.png)
[https://i.imgur.com/46C04hs.png](https://i.imgur.com/46C04hs.png)

#### Output
[https://i.imgur.com/0h3Y9Ml.png](https://i.imgur.com/0h3Y9Ml.png)
[https://i.imgur.com/46C04hs.png](https://i.imgur.com/46C04hs.png)

---
```