

## EXPERIMENT NUMBER –Practical 6.1

STUDENT'S NAME –

STUDENT'S UID -

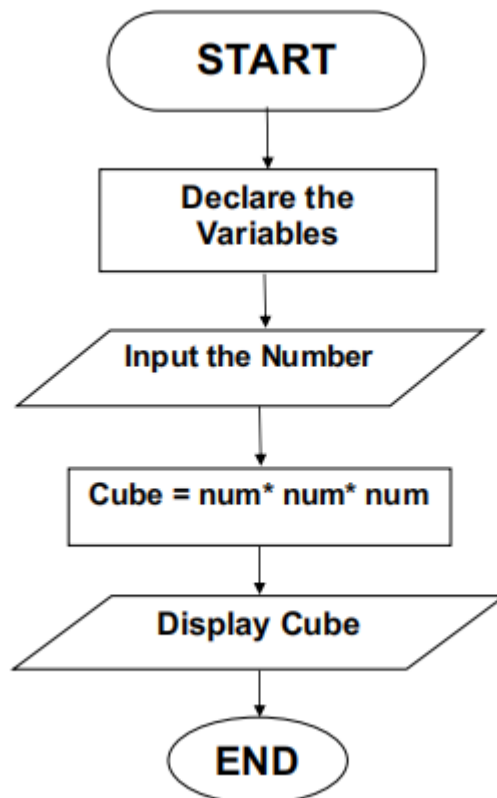
CLASS AND GROUP-

SEMESTER - 2

TOPIC OF EXPERIMENT - POLYMORPHISM

AIM OF THE EXPERIMENT - WAP to calculate and display cube of an integer and float variable using function overloading.

FLOWCHART/ALGORITHM-



## PROGRAM CODE-

```
9 #include <iostream>
10
11 using namespace std;
12
13 int cube(int );
14
15 float cube(float);
16
17 int main() {
18     int a = 7;
19     float b = 7.5;
20
21     cout<< "Cube of integer number " << a << " is " << cube(a) <<endl;
22
23     cout<< "Cube of float number " << b << " is " << cube(b) <<endl;
24
25     return 0;
26 }
27
28
29
30
31 int cube(int x) {
32     return x*x*x;
33 }
34
35
36
37 float cube(float y){
38     return y*y*y;
39 }
40
41 }
```

## OUTPUT -

```
Cube of integer number 7 is 343
Cube of float number 7.5 is 421.875

...Program finished with exit code 0
Press ENTER to exit console.█
```

## EXPERIMENT NUMBER –Practical 6.2

STUDENT'S NAME - YASH RAJ

STUDENT'S UID - 21BCS11765

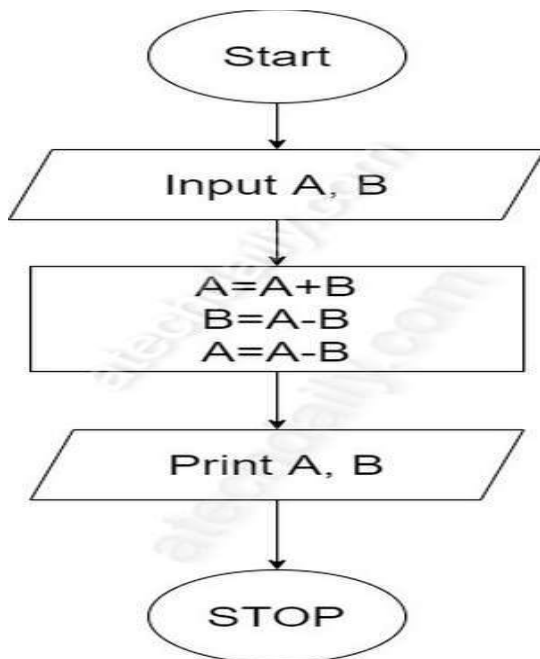
CLASS AND GROUP- 509-B

SEMESTER - 2

### TOPIC OF EXPERIMENT – POLYMORPHISM

**AIM OF THE EXPERIMENT** - Program to demonstrate the unary operator overloading for operator ++. Make a class test. Create a default constructor to initialize the variable. 1) Overload operator ++ (Pre) with definition to pre-decrement the value of a variable 2) Overload operator ++ (post) with definition to post-decrement the value of variable.

### FLOWCHART/ALGORITHM -



## PROGRAM CODE-

```

1 #include <iostream>
2
3 using namespace std;
4
5 class Test {
6 private:
7     int num;
8
9 public:
10    // required constructors
11    // default constructor to initialize the variable
12    Test() {
13        num = 0;
14    }
15    // parameterized constructor to return object after incrementing
16    Test(int n) {
17        num = n;
18    }
19
20    // method to display time
21    void display() {
22        cout << "Number: " << num << endl;
23    }
24
25    // overloaded prefix ++ operator
26    Test operator++ () {

```

```

27    // overloaded prefix ++ operator
28    Test operator++ () {
29        // increment this object
30        ++num;
31
32        // return object with increment value
33        return Test(num);
34    }
35
36    // overloaded postfix ++ operator
37    Test operator++ (int ) {
38        // save the original value
39        Test t(num);
40
41        // increment current object
42        ++num;
43
44        // return old original value
45        return t;
46    }
47
48    int main() {
49        Test T1(1), T2(1), T3;

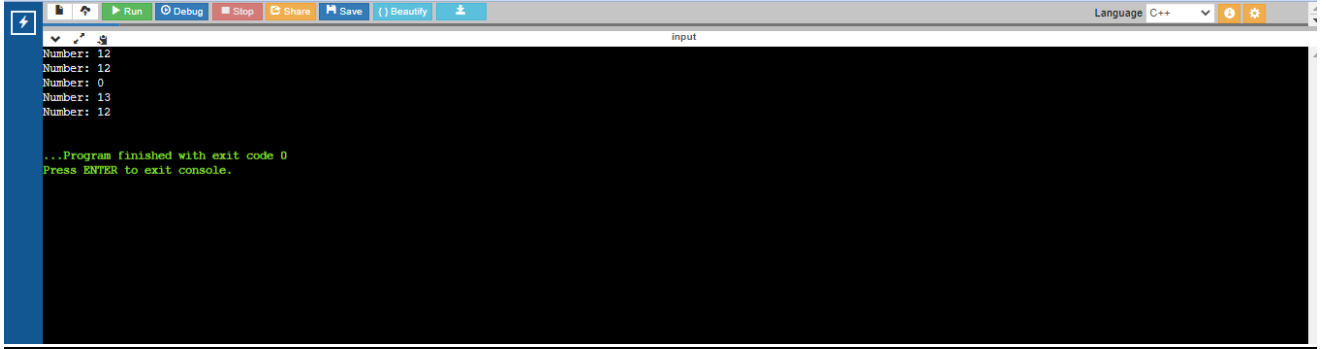
```

```

50        // save the original value
51        Test t(num);
52
53        // increment current object
54        ++num;
55
56        // return old original value
57        return t;
58    }
59
60    int main() {
61        Test T1(1), T2(1), T3;
62
63        ++T1; // increment T1
64        T1.display(); // display T1
65
66        T2++; // increment T2
67        T2.display(); // display T2
68
69        T3.display(); // display T3
70
71        T3 = T2++; // increment T2 again and assign pre-incremented value to T3
72        T2.display(); // display T2
73        T3.display(); // display T3
74
75        return 0;
76    }

```

## OUTPUT-



```
Number: 12
Number: 12
Number: 0
Number: 13
Number: 12

...Program finished with exit code 0
Press ENTER to exit console.
```

## EXPERIMENT NUMBER – Practical 6.3

STUDENT'S NAME – YASH RAJ

STUDENT'S UID – 21BCS11765

CLASS AND GROUP – 509-B

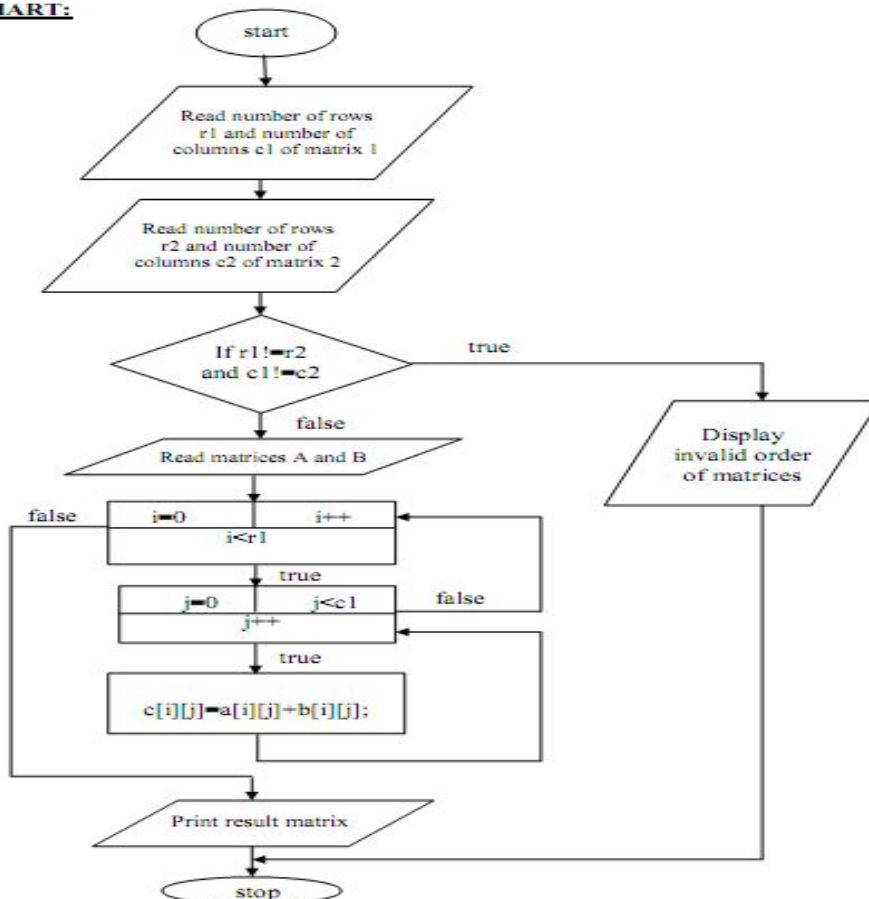
SEMESTER – 2ND

### TOPIC OF EXPERIMENT – POLYMORPHISM

**AIM OF THE EXPERIMENT** - WAP for creating a matrix class which can handle integer matrices of different dimensions. Overload the operator (+) for addition and (==) comparison of matrices.

### FLOWCHART/ ALGORITHM -

#### FLOWCHART:



## PROGRAM CODE-

```

1 #include <iostream>
2
3 #define MAXROWS 50
4 #define MAXCOLS 50
5
6 using namespace std;
7
8 // Class for Matrix operator overloading
9
10 class Matrix {
11 public:
12     // For input Matrix
13     int arr[MAXROWS][MAXCOLS];
14     int rows, cols;
15
16 Matrix() {
17     rows = cols = 0;
18 }
19
20 // Overload constructor to initialize the Matrix with dimensions
21 Matrix(int r, int c, int mat[MAXROWS][MAXCOLS]) {
22     rows = r;
23     cols = c;
24
25     for (int i = 0; i < rows; i++) {
26         for (int j = 0; j < cols; j++) {
27             arr[i][j] = mat[i][j];
28         }
29     }
30 }
31
32 // Function to display the elements of Matrix
33 void display() {
34     for (int i = 0; i < rows; i++) {
35         cout << " ";
36
37         for (int j = 0; j < cols; j++) {
38             // Print the element
39             cout << arr[i][j] << " ";
40         }
41         cout << "\n" << endl;
42     }
43 }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

```

59     // Print the element
60     cout << arr[i][j] << " ";
61 }
62 }
63
64     cout << "\n" << endl;
65 }
66 }
67
68     cout << endl;
69 }
70 }
71
72
73
74
75 // Function for + operator overloading
76 Matrix operator+(Matrix x) {
77     if (x.rows != rows || x.cols != cols || (rows == 0 || cols == 0)) {
78         return Matrix();
79     }
80
81     // To store the sum of Matrices
82     int mat[MAXROWS][MAXCOLS];
83
84     // Traverse the Matrix x
85     for (int i = 0; i < rows; i++) {
86         for (int j = 0; j < cols; j++) {
87             // Add the corresponding blocks of Matrices
88             mat[i][j] = arr[i][j] + x.arr[i][j];
89         }
90     }
91     return Matrix(rows, cols, mat);
92 }
93
94 // Function for -= operator overloading
95 int operator==(Matrix a) {
96     if (x.rows != rows || x.cols != cols) {
97         return 0;
98     }
99 }
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

```

```

176 arr2[1][1] = 1;
177
178
179
180
181 // Declare Matrices
182
183 Matrix mat1(2, 2, arr1);
184
185 Matrix mat2(2, 2, arr2);
186
187 Matrix mat3(2, 2, arr3);
188
189 Matrix mat4;
190
191
192 // Printing the elements of first matrix
193
194 cout<< "Elements of Matrix 1:" <<endl;
195
196 mat1.display();
197
198
199 // Printing the elements of second matrix
200
201 cout<< "Elements of Matrix 2:" <<endl;
202
203 mat2.display();
204
205
206 // Printing the elements of third matrix
207
208 cout<< "Elements of Matrix 3:" <<endl;
209
210 mat3.display();
211
212
213 // Addition of two matrices using operator overloading
214
215 mat4 = mat1 + mat3;
216
217 cout<< "Elements of Matrix after addition of Matrix 1 and Matrix 3:" <<endl;
218
219 mat4.display();
220
221
222 // Equating two matrices using operator overloading
223
224 if (mat1 == mat2) {
225
226 cout<< "Matrix 1 is equals to Matrix 2" <<endl;
227
228 }
229
230 else {
231
232 cout<< "Matrix 1 is not equals to Matrix 2" <<endl;
233
234 }
235
236
237
238
239
240

```

```

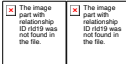
241 // Printing the elements of second matrix
242
243 cout<< "Elements of Matrix 2:" <<endl;
244
245 mat2.display();
246
247
248 // Printing the elements of third matrix
249
250 cout<< "Elements of Matrix 3:" <<endl;
251
252 mat3.display();
253
254
255 // Addition of two matrices using operator overloading
256
257 mat4 = mat1 + mat3;
258
259 cout<< "Elements of Matrix after addition of Matrix 1 and Matrix 3:" <<endl;
260
261 mat4.display();
262
263
264 // Equating two matrices using operator overloading
265
266 if (mat1 == mat2) {
267
268 cout<< "Matrix 1 is equals to Matrix 2" <<endl;
269
270 }
271
272 else {
273
274 cout<< "Matrix 1 is not equals to Matrix 2" <<endl;
275
276 }
277
278
279
280
281 // Equating two matrices using operator overloading
282
283 if (mat1 == mat3) {
284
285 cout<< "Matrix 1 is equals to Matrix 3" <<endl;
286
287 }
288
289 else {
290
291 cout<< "Matrix 1 is not equals to Matrix 3" <<endl;
292
293 }
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



## OUTPUT-

```
Elements of Matrix 1:  
[ 1, 2, ]  
[ 3, 4, ]  
  
Elements of Matrix 2:  
[ 1, 2, ]  
[ 3, 4, ]  
  
Elements of Matrix 3:  
[ 4, 3, ]  
[ 2, 1, ]  
  
Elements of Matrix after addition of Matrix 1 and Matrix 3:  
[ 5, 5, ]  
[ 5, 5, ]  
  
Matrix 1 is equals to Matrix 2  
Matrix 1 is not equals to Matrix 3
```



## EXPERIMENT NUMBER –Practical 6.4

STUDENT'S NAME – YASH RAJ

STUDENT'S UID – 21BCS11765

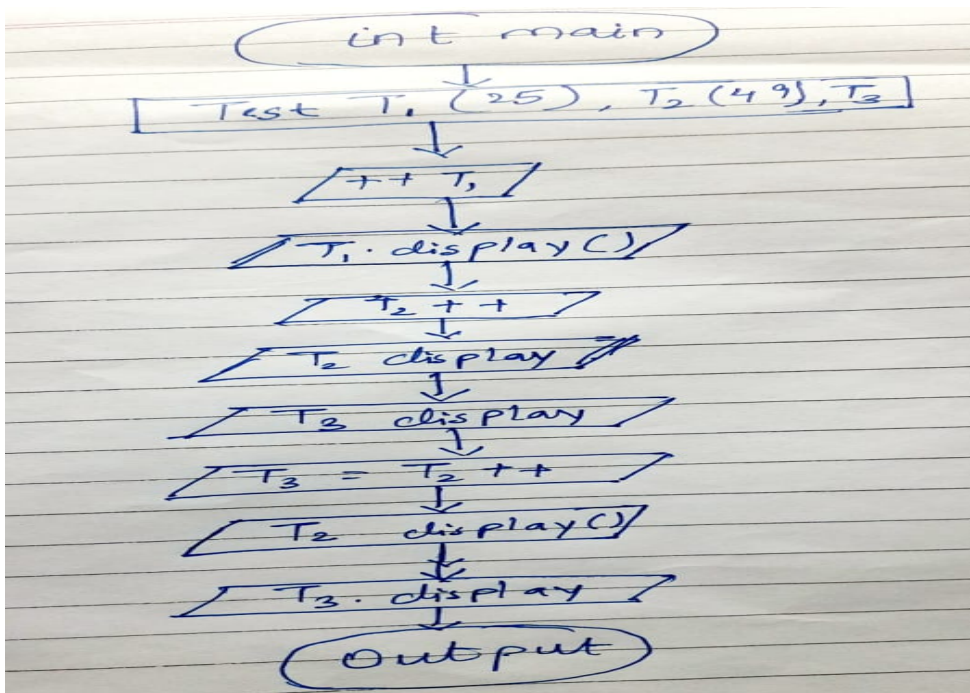
CLASS AND GROUP – 509-B

SEMESTER –2ND

### TOPIC OF EXPERIMENT - POLYMORPHISM

AIM OF THE EXPERIMENT - WAP to create a class Pairs. Objects of type Pairs can be used in any situation where ordered pairs are needed. Our Task is to overload operator >> and << so that objects of class Pairs are to be input and output in the form (5,3) (5,-6) (-5,6) or (-5,-3). There is no need to implement any constructor/method .

### FLOWCHART/ALGORITHM-



## PROGRAM CODE-

```
8
9 #include <iostream>
10
11 using namespace std;
12
13 class Test {
14
15 private:
16
17 int num;
18
19
20
21 public:
22
23 Test() {
24
25 num = 0;
26
27 }
28
29 Test(int n) {
30
31 num = n;
32
33 }
34
35 void display() {
36
37 cout<< "Number: " <<num<<endl;
38
39 }
40
41 Test operator++ () {
42
43 ++num;
44
45 return Test(num);
46
47 }
48
49 Test operator++( int ) {
50
51 Test t(num);
52 ++num;
53
54 return t;
55
56 }
57
58 };
59
60
61 int main() {
62
63 Test T1(25), T2(49), T3;
64
65 ++T1;
66 T1.display();
67
68 T2++;
```

## OUTPUT-

```
Number: 26
Number: 50
Number: 0
Number: 51
Number: 50

..Program finished with exit code 0
Press ENTER to exit console.□
```

### LEARNING OUTCOMES

- Identify situations where computational methods would be useful.
- Approach the programming tasks using techniques learnt and write pseudo-code.
- Choose the right data representation formats based on the requirements of the problem.
- Use the comparisons and limitations of the various programming constructs and choose the right one for the task.

### EVALUATION COLUMN (To be filled by concerned faculty only)

Sr. No.	Parameters	Maximum Marks	Marks Obtained
1.	Worksheet Completion including writing learning objective/ Outcome	10	
2.	Post Lab Quiz Result	5	
3.	Student engagement in Simulation/ Performance/ Pre Lab Questions	5	
4.	Total Marks	20	



**CHANDIGARH  
UNIVERSITY**  
Discover. Learn. Empower.

**SUBJECT NAME-** OBJECT ORIENTED PROGRAMMING  
USING C++ LAB

**SUBJECT CODE-**21CSH-  
103