

~\Desktop\Data Structures.cpp

```
1  عمرو حمزه عمرو محمد عظيمي //
2  #include<iostream>
3  #include <cstdlib>
4  using namespace std;
5  int main();
6  void Dynamic_insert_type();
7  void static_insert_type();
8  void dynamicMenu();
9  void myStack();
10 void stack_print();
11 void stack_pop ();
12 int stack_push (int val);
13 int stack_isempty();
14 void myQueue();
15 void queuePrint();
16 void queuePop();
17 void queuePush(int x);
18 int queueIsempy();
19 void myList();
20 void insertAtBeginning(int value);
21 void insertAtEnd(int value);
22 void deleteValue(int value);
23 void deleteFirstNode();
24 void deleteLastNode();
25 void insertBeforeNode(int beforeValue, int newValue);
26 void insertAfterNode(int afterValue, int newValue);
27 void updateNode(int oldVal, int newVal);
28 void sortList();
29 void displayList();
30
31
32 ///////////////////////////////////////////////////////////////////
33 struct stack1 {
34     int data;
35     stack1* last;
36 };
37 stack1 *top = NULL , *newTop=NULL;
38
39 struct queue1 {
40     int data;
41     queue1* next;
42 };
43 queue1 *queueHead=NULL ,*queueTail=NULL, *newQueue=NULL;
44
45 struct Node {
46     int data;
47     Node* next;
48 };
49 Node* head=NULL , *newNode=NULL;
50
51 int insert_type_val = 0;
52 int static_insert_type_val = 0;
53 ///////////////////////////////////////////////////////////////////
54
55 void dynamicMenu()
56 {
57     system("cls");
```

```

58     int select;
59     cout << "\n";
60     cout << "                Enter The Type Of Structures \n\n";
61     cout << "                ----- \n";
62     cout << "                |      (Dynamic Structures)      | \n";
63     cout << "                ----- \n";
64     cout << "                | 1: Stack                          | \n";
65     cout << "                | 2: Queue                          | \n";
66     cout << "                | 3: List                          | \n";
67     cout << "                ----- \n\n";
68     cout << "Press 0 to go back\n\n";
69     cout << "Enter The Structures Number -> ";
70     cin>>select;
71     switch(select)
72     {
73         case 1:
74             insert_type_val = 1;
75             Dynamic_insert_type();
76             break;
77         case 2:
78             insert_type_val = 2;
79             Dynamic_insert_type();
80             break;
81         case 3:
82             insert_type_val = 3;
83             Dynamic_insert_type();
84             break;
85         case 0:
86             main();
87             break;
88         default:
89             dynamicMenu();
90             break;
91     }
92
93 }
94
95 void Dynamic_insert_type()
96 {
97
98
99
100     system("cls");
101     int select;
102     cout << "\n";
103     cout << "                ----- \n";
104     cout << "                |      (Choose)      | \n";
105     cout << "                ----- \n";
106     cout << "                | 1: Manual Enter      | \n";
107     cout << "                | 2: Random            | \n";
108     cout << "                ----- \n\n";
109     cout << "Press 0 to go back\n\n";
110     cout << "Enter The Number -> ";
111     cin>>select;
112     switch(select)
113     {
114         case 1:
115             if(insert_type_val == 1)
116                 myStack();
117             else if(insert_type_val == 2)

```

```

118         myQueue();
119     else if(insert_type_val == 3)
120         myList();
121 case 2:
122     if(insert_type_val == 1)
123     {
124         for (int i = 0; i < 7; i++)
125         {
126             stack_push(rand()%10+1);
127         }
128         myStack();
129     }
130
131     if(insert_type_val == 2)
132     {
133         for (int i = 0; i < 7; i++)
134         {
135             queuePush(rand()%10+1);
136         }
137         myQueue();
138     }
139
140     if(insert_type_val == 3)
141     {
142         for (int i = 0; i < 7; i++)
143         {
144             insertAtEnd(rand()%10+1);
145         }
146         myList();
147     }
148     break;
149 case 0:
150     system("cls");
151     dynamicMenu();
152     break;
153 default:
154     system("cls");
155     Dynamic_insert_type();
156     break;
157 }
158 }
159 //////////////////////////////////////////////////Stack Start////////////////////////////////////
160
161 void myStack()
162 {
163     system("cls");
164     int op , val;
165     cout << "        ---Stack---\n\n";
166     cout << "        Content Of Stack\n\n";
167     stack_print();
168     cout << "*****\n";
169     cout << "The Operation Of Stack\n\n";
170     cout << "  1->Push\n";
171     cout << "  2->Pop\n\n";
172     cout << "Press 0 to go back\n\n";
173     cout<<"Insert The Operation Number -> ";
174     cin >> op;
175
176     switch(op) {
177         case 1:

```

```

178         cout<<"\nInsert The Value : ";
179         cin>> val;
180         stack_push(val);
181         cout<<"\n";
182         myStack();
183         break;
184
185     case 2:
186         stack_pop();
187         cout<<"\n";
188         myStack();
189         break;
190
191     case 0:
192         dynamicMenu();
193         break;
194     default:
195         cout << "Incorrect Operation\n\n";
196         system("pause");
197         myStack();
198         break;
199 }
200 }
201
202 int stack_isempty()
203 {
204     if(top == NULL)
205         return 0;
206     else
207         return 1;
208 }
209 int stack_push (int val)
210 {
211     newTop=new stack1;
212     newTop->data = val;
213     newTop->last = NULL;
214
215     if(top != NULL)
216         newTop->last = top;
217
218     top = newTop;
219
220 }
221 void stack_pop ()
222 {
223     if(stack_isempty())
224     {
225         top = newTop->last;
226         cout<<"\n";
227         cout<<"Delete-> "<<newTop->data<<"\n"<< endl;
228         delete newTop;
229         newTop = top;
230     }
231     else
232     {
233         cout<<"\n";
234         cout<<"Stack Is Empty\n\n";
235     }
236     system("pause");
237

```

```

238 }
239 void stack_print()
240 {
241     if (!stack_isempty())
242     {
243         cout << "\n      Stack is Empty\n\n";
244     }
245
246     stack1 *temp = top;
247     while (temp != NULL)
248     {
249         cout << "          "<< temp->data << " \n";
250         temp = temp->last;
251     }
252     cout << "\n";
253
254 }
255 //////////////////////////////////////////////////Stack End//////////////////////////////////////
256
257
258 //////////////////////////////////////////////////Queue Start//////////////////////////////////////
259 void myQueue()
260 {
261     int op, val;
262     system("cls");
263     cout << "          ---Queue---\n\n";
264     cout << "      Content Of Queue\n\n";
265     queuePrint();
266     cout << "*****\n";
267     cout << "The Operation Of Queue\n\n";
268     cout << "  1->Push\n";
269     cout << "  2->Pop\n\n";
270     cout << "Press 0 to go back\n\n";
271     cout<<"Insert The Operation Number -> ";
272     cin >> op;
273
274     switch(op)
275     {
276         case 1:
277             cout << "\nEnter the Value: ";
278             cin >> val;
279             queuePush(val);
280             cout<<"\n";
281             myQueue();
282             break;
283         case 2:
284             queuePop();
285             cout<<"\n";
286             myQueue();
287             break;
288         case 0:
289             dynamicMenu();
290             break;
291         default:
292             cout << "Incorrect Operation\n";
293             system("pause");
294             myQueue();
295             break;
296     }
297 }

```

```

298 int queueIsEmpty()
299 {
300     if (queueHead == NULL)
301         return 0;
302     else
303         return 1;
304 }
305 void queuePush(int x)
306 {
307     newQueue = new queue1;
308     newQueue->data = x;
309     newQueue->next = NULL;
310     if (queueHead == NULL)
311         queueHead = queueTail = newQueue;
312     else
313     {
314         queueTail->next = newQueue;
315         queueTail = newQueue;
316     }
317 }
318 void queuePop()
319 {
320     if (queueIsEmpty())
321     {
322         queue1 *p = queueHead;
323         cout << "\nDelete-> " << queueHead->data << endl << endl;
324         queueHead = queueHead->next;
325         delete p;
326     }
327     else
328         cout << "\nQueue Is Empty\n\n";
329     system("pause");
330 }
331 void queuePrint()
332 {
333     if (!queueIsEmpty())
334     {
335         cout << "\n      Queue Is Empty\n\n";
336     }
337     queue1 *temp = queueHead;
338     while (temp != NULL)
339     {
340         cout << " " << temp->data << " ";
341         temp = temp->next;
342     }
343     cout << "\n\n";
344 }
345 //////////////////////////////////////////////////Queue End////////////////////////////////////
346
347
348 //////////////////////////////////////////////////List Start////////////////////////////////////
349 void myList()
350 {
351     int op, val, oldVal, newVal, refVal;
352     system("cls");
353     cout << "      ---List---\n\n";
354     cout << "      Content Of List\n\n";
355     displayList();
356     cout << "*****\n";
357     cout << "The Operation Of List\n\n";

```

```

358 cout << " 1->Insert value\n";
359 cout << " 2->Insert at Beginning\n";
360 cout << " 3->Insert at End\n";
361 cout << " 4->Insert Before Node\n";
362 cout << " 5->Insert After Node\n";
363 cout << " 6->Delete value from any\n";
364 cout << " 7->Delete the first value\n";
365 cout << " 8->Delete the Last value\n";
366 cout << " 9->Update Node\n";
367 cout << " 10->Sort List\n\n";
368 cout << "Press 0 to go back\n\n";
369 cout<<"Insert The Operation Number -> ";
370 cin >> op;
371
372 switch (op) {
373 case 1:
374     cout << "\nEnter the value-> ";
375     cin >> val;
376     insertAtEnd(val);          //Insert value
377     myList();
378     break;
379 case 2:
380     cout << "\nEnter the value-> ";
381     cin >> val;
382     insertAtBeginning(val);    //Insert at Beginning
383     myList();
384     break;
385 case 3:
386     cout << "\nEnter the value-> ";
387     cin >> val;
388     insertAtEnd(val);          //Insert at End
389     myList();
390     break;
391 case 4:
392     cout << "\nEnter the Node value-> ";
393     cin >> refVal;
394     cout << "\nEnter the value to insert: ";
395     cin >> val;
396     insertBeforeNode(refVal, val);    //Insert Before Node
397     myList();
398     break;
399 case 5:
400     cout << "\nEnter the Node value-> ";
401     cin >> refVal;
402     cout << "\nEnter the value to insert-> ";
403     cin >> val;
404     insertAfterNode(refVal, val);    //Insert After Node
405     myList();
406     break;
407 case 6:
408     cout << "\nEnter the value to delete-> ";
409     cin >> val;
410     deleteValue(val);            //Delete value from any
411     myList();
412     break;
413 case 7:
414     deleteFirstNode();           //Delete the first value
415     myList();
416     break;
417 case 8:

```

```

418     deleteLastNode();          //Delete the Last value
419     myList();
420     break;
421 case 9:
422     cout << "\nEnter the old value-> ";
423     cin >> oldVal;
424     cout << "\nEnter the new value-> ";
425     cin >> newVal;
426     updateNode(oldVal, newVal);    //Update Node
427     myList();
428     break;
429 case 10:
430     sortList();                  //Sort List
431     myList();
432     break;
433 case 0:
434     dynamicMenu();              //return to Main Menu
435     break;
436 default:
437     cout << "Incorrect Operation\n";
438     myList();
439     break;
440 }
441 }
442
443 void insertAtBeginning(int value)
444 {
445     Node *newNode = new Node();
446     newNode->data = value;
447     newNode->next = head;
448     head = newNode;
449 }
450
451 void insertAtEnd(int value)
452 {
453     Node* newNode = new Node();
454     newNode->data = value;
455     newNode->next = NULL;
456
457     if (head == NULL)
458     {
459         head = newNode;
460     }
461     else
462     {
463         Node* temp = head;
464         while (temp->next != NULL)
465         {
466             temp = temp->next;
467         }
468         temp->next = newNode;
469     }
470 }
471
472 void insertBeforeNode(int beforeValue, int newValue)
473 {
474     Node* newNode = new Node();
475     newNode->data = newValue;
476
477     if (head == NULL)

```



```

478     {
479         cout << "\nThe List Is Empty\n\n";
480         system("pause");
481         delete newNode;
482         return;
483     }
484
485     if (head->data == beforeValue)
486     {
487         newNode->next = head;
488         head = newNode;
489         return;
490     }
491
492     Node* temp = head;
493     while (temp->next != NULL && temp->next->data != beforeValue)
494     {
495         temp = temp->next;
496     }
497
498     if (temp->next == NULL)
499     {
500         cout << "\nValue Not Found In The List\n\n";
501         system("pause");
502         delete newNode;
503         return;
504     }
505
506     newNode->next = temp->next;
507     temp->next = newNode;
508 }
509
510 void insertAfterNode(int afterValue, int newValue)
511 {
512     Node* newNode = new Node();
513     newNode->data = newValue;
514
515     Node* temp = head;
516     while (temp != NULL && temp->data != afterValue)
517     {
518         temp = temp->next;
519     }
520
521     if (temp == NULL)
522     {
523         cout << "\nValue Not Found In The List\n\n";
524         system("pause");
525         delete newNode;
526         return;
527     }
528
529     newNode->next = temp->next;
530     temp->next = newNode;
531 }
532
533 void deleteFirstNode()
534 {
535     if (head == NULL)
536     {
537         cout << "\nThe List Is Empty\n\n";

```

```

538         system("pause");
539         return;
540     }
541
542     Node* temp = head;
543     head = head->next;
544     delete temp;
545 }
546
547 void deleteLastNode()
548 {
549     if (head == NULL)
550     {
551         cout << "\nThe List Is Empty\n\n";
552         system("pause");
553         return;
554     }
555
556     if (head->next == NULL)
557     {
558         delete head;
559         head = NULL;
560         return;
561     }
562
563     Node* temp = head;
564     while (temp->next->next != NULL)
565     {
566         temp = temp->next;
567     }
568
569     delete temp->next;
570     temp->next = NULL;
571 }
572
573 void deleteValue(int value)
574 {
575     Node* temp = head;
576     Node* prev = NULL;
577
578     if (temp != NULL && temp->data == value)
579     {
580         head = temp->next;
581         delete temp;
582         return;
583     }
584
585     while (temp != NULL && temp->data != value)
586     {
587         prev = temp;
588         temp = temp->next;
589     }
590
591     if (temp == NULL) return;
592
593     prev->next = temp->next;
594     delete temp;
595 }
596
597 void updateNode(int oldVal, int newVal)

```

```

598 {
599     Node* temp = head;
600     while (temp != NULL)
601     {
602         if (temp->data == oldVal)
603         {
604             temp->data = newVal;
605             return;
606         }
607         temp = temp->next;
608     }
609     cout << "\nValue Not Found In The List\n\n";
610     system("pause");
611 }
612
613 void sortList()
614 {
615     if (head == NULL || head->next == NULL)
616     {
617         return;
618     }
619
620     Node* i = head;
621     Node* j = NULL;
622     int temp;
623
624     while (i != NULL)
625     {
626         j = i->next;
627         while (j != NULL) {
628             if (i->data > j->data)
629             {
630                 temp = i->data;
631                 i->data = j->data;
632                 j->data = temp;
633             }
634             j = j->next;
635         }
636         i = i->next;
637     }
638 }
639
640 void displayList()
641 {
642     Node *temp = head;
643     while (temp != NULL)
644     {
645         cout << temp->data << " -> ";
646         temp = temp->next;
647     }
648     cout << "NULL\n\n";
649 }
650 //////////////////////////////////////////////////List End////////////////////////////////////
651
652
653 //////////////////////////////////(static Structures)////////////////////////////////////
654 int* static_arr;
655 int static_top = -1;
656 int sizes;
657 int h = -1;

```

```

658 int t = -1;
659 int c = 0;
660 //*****
661 void my_Static_Stack();
662 void Static_Stack_pop();
663 void Static_Stack_push(int val);
664 int Static_Stack_isempty();
665 int Static_Stack_isfull();
666 void stack_Stack_print();
667 //*****
668 void my_Static_Liner_Queue();
669 void Static_Liner_Queue_Push(int val);
670 void Static_Liner_Queue_Pop();
671 int Static_Liner_Queue_isempty();
672 int Static_Liner_Queue_isfull();
673 void queue_print();
674 //*****
675 void my_Static_Circular_Queue();
676 void Static_Circular_Queue_Push(int val);
677 void Static_Circular_Queue_Pop();
678 int Static_Circular_Queue_isempty();
679 int Static_Circular_Queue_isfull();
680 void Circular_queue_print();
681 //*****
682 void staticMenu()
683 {
684     system("cls");
685     int select;
686     cout << "\n";
687     cout << "                Enter The Type Of Structures \n\n";
688     cout << "                ----- \n";
689     cout << "                |      (static Structures)      | \n";
690     cout << "                ----- \n";
691     cout << "                | 1: Stack                        | \n";
692     cout << "                | 2: Liner Queue                  | \n";
693     cout << "                | 3: Circular Queue               | \n";
694     cout << "                ----- \n\n";
695     cout << "Press 0 to go back\n\n";
696     cout << "Enter The Transaction Number -> ";
697     cin>>select;
698     switch(select)
699     {
700     case 1:
701         static_insert_type_val = 1;
702         static_insert_type();
703         break;
704     case 2:
705         static_insert_type_val = 2;
706         static_insert_type();
707         break;
708     case 3:
709         static_insert_type_val = 3;
710         static_insert_type();
711         break;
712     case 0:
713         main();
714         break;
715     default:
716         staticMenu();
717         break;

```

```

718     }
719 }
720
721 void static_insert_type()
722 {
723
724
725
726     system("cls");
727     int select;
728     cout << "\n";
729     cout << "          ----- \n";
730     cout << "          |          (Choose)          | \n";
731     cout << "          ----- \n";
732     cout << "          | 1: Manual Enter          | \n";
733     cout << "          | 2: Random                | \n";
734     cout << "          ----- \n\n";
735     cout << "Press 0 to go back\n\n";
736     cout << "Enter The Number -> ";
737     cin>>select;
738     switch(select)
739     {
740     case 1:
741         if(static_insert_type_val == 1)
742         {
743             system("cls");
744             cout << "Enter the size of the stack-> ";
745             cin >> sizes;
746             static_arr = new int[sizes];
747             static_top = -1;
748             my_Static_Stack();
749         }
750
751
752
753         else if(static_insert_type_val == 2)
754         {
755             system("cls");
756             cout << "Enter the size of the Linear queue-> ";
757             cin >> sizes;
758             static_arr = new int[sizes];
759             h = -1;
760             t = -1;
761             my_Static_Linear_Queue();
762         }
763
764
765         else if(static_insert_type_val == 3)
766         {
767             system("cls");
768             cout << "Enter the size of the Circular queue-> ";
769             cin >> sizes;
770             static_arr = new int[sizes];
771             h = -1;
772             t = -1;
773             c = 0;
774             my_Static_Circular_Queue();
775         }
776
777

```

```

778     case 2:
779         if(static_insert_type_val == 1)
780         {
781             system("cls");
782             cout << "Enter the size of the stack-> ";
783             cin >> sizes;
784             static_arr = new int[sizes];
785             static_top = -1;
786             for (int i = 0; i < sizes; i++)
787             {
788                 Static_Stack_push(rand()%10+1);
789             }
790             my_Static_Stack();
791         }
792
793         if(static_insert_type_val == 2)
794         {
795             system("cls");
796             cout << "Enter the size of the Linear queue-> ";
797             cin >> sizes;
798             static_arr = new int[sizes];
799             h = -1;
800             t = -1;
801             for (int i = 0; i < sizes; i++)
802             {
803                 Static_Liner_Queue_Push(rand()%10+1);
804             }
805             my_Static_Liner_Queue();
806         }
807
808         if(static_insert_type_val == 3)
809         {
810             system("cls");
811             cout << "Enter the size of the Circular queue-> ";
812             cin >> sizes;
813             static_arr = new int[sizes];
814             h = -1;
815             t = -1;
816             c = 0;
817             for (int i = 0; i < sizes; i++)
818             {
819                 Static_Circular_Queue_Push(rand()%10+1);
820             }
821             my_Static_Circular_Queue();
822         }
823
824         break;
825     case 0:
826         system("cls");
827         staticMenu();
828         break;
829     default:
830         static_insert_type();
831         break;
832 }
833 }
834 //*****
835
836 void my_Static_Stack()
837 {

```

```

838     system("cls");
839     int op , val;
840     cout << "    ---Stack size Is "<<size<<"---\n\n";
841     stack_Stack_print();
842     cout << "*****\n";
843     cout << "The Operation Of Stack\n\n";
844     cout << "    1->Push\n";
845     cout << "    2->Pop\n\n";
846     cout << "Press 0 to go back\n\n";
847     cout<<"Insert The Operation Number-> ";
848     cin >> op;
849
850     switch(op) {
851         case 1:
852             if(Static_Stack_isfull()==0)
853             {
854                 cout<<"\nInsert The Value : ";
855                 cin>> val;
856                 Static_Stack_push(val);
857                 cout<<"\n";
858             }
859             else
860             {
861                 cout << "\n";
862                 cout<<"Stack Is Full\n";
863                 cout << "\n\n";
864                 system("pause");
865             }
866
867             my_Static_Stack();
868             break;
869
870         case 2:
871             if(Static_Stack_isempty()==0)
872                 Static_Stack_pop();
873             else
874             {
875                 cout << "\n";
876                 cout<<"Stack Is Empty\n";
877                 cout << "\n\n";
878                 system("pause");
879             }
880             my_Static_Stack();
881             break;
882
883         case 0:
884             if(Static_Stack_isempty()==0)
885             {
886                 for (int i = 0; i <= static_top; i++)
887                 {
888                     static_arr[static_top--];
889                 }
890             }
891
892             staticMenu();
893             break;
894         default:
895             cout << "\n";
896             cout << "Incorrect Operation\n\n";
897

```

```

898         cout << "\n\n";
899         system("pause");
900         my_Static_Stack();
901         break;
902     }
903 }
904
905 int Static_Stack_isfull()
906 {
907     if (static_top == sizes - 1)
908         return 1;
909     else
910         return 0;
911 }
912
913 int Static_Stack_isempty()
914 {
915     if (static_top == -1)
916         return 1;
917     else
918         return 0;
919 }
920
921 void Static_Stack_push(int val)
922 {
923     static_arr[++static_top] = val;
924 }
925
926 void Static_Stack_pop()
927 {
928     cout << "\n";
929     cout << "Delete: " << static_arr[static_top--] << "\n";
930     cout << "\n";
931     system("pause");
932     cout << "\n\n";
933 }
934
935 void stack_Stack_print()
936 {
937
938     if (Static_Stack_isempty())
939     {
940         cout << "        Stack Is Empty\n\n";
941     }
942     else
943     {
944         for (int i = static_top; i >= 0; i--)
945         {
946             cout << "                " << static_arr[i] << "\n";
947         }
948         cout << "\n";
949     }
950 }
951 //*****
952
953 void my_Static_Liner_Queue()
954 {
955     system("cls");
956     int op, val;
957     cout << " ---Liner Queue size Is "<<sizes<<"---\n\n";

```



```

958 queue_print();
959 cout << "*****\n";
960 cout << "The Operation Of Liner Queue\n\n";
961 cout << " 1->Push\n";
962 cout << " 2->Pop\n\n";
963 cout << "Press 0 to go back\n\n";
964 cout << "Insert The Operation Number-> ";
965 cin >> op;
966
967 switch (op)
968 {
969     case 1:
970         if (Static_Liner_Queue_isfull() == 0)
971         {
972             cout << "\nInsert The Value : ";
973             cin >> val;
974             Static_Liner_Queue_Push(val);
975             cout << "\n";
976         }
977         else
978         {
979             cout << "\n";
980             cout << "Queue Is Full\n";
981             cout << "\n\n";
982             system("pause");
983         }
984
985         my_Static_Liner_Queue();
986         break;
987
988     case 2:
989         if (Static_Liner_Queue_isempty() == 0)
990             Static_Liner_Queue_Pop();
991         else
992         {
993             cout << "\n";
994             cout << "Queue Is Empty\n";
995             cout << "\n\n";
996             system("pause");
997         }
998         my_Static_Liner_Queue();
999         break;
1000
1001     case 0:
1002         for(int i= 0; i<=c; i++)
1003             static_arr[h++];
1004         h = -1;
1005         t = -1;
1006         c = 0;
1007         staticMenu();
1008         break;
1009     default:
1010         cout << "\n";
1011         cout << "Incorrect Operation\n\n";
1012         cout << "\n\n";
1013         system("pause");
1014         my_Static_Liner_Queue();
1015         break;
1016 }
1017 }

```

```

1018
1019 void Static_Liner_Queue_Push(int val)
1020 {
1021     if(h==-1)
1022         h=0;
1023     static_arr[++t] = val;
1024     c++;
1025 }
1026
1027 void Static_Liner_Queue_Pop()
1028 {
1029     if (Static_Liner_Queue_isempty() == 1)
1030     {
1031         cout << "\nQueue Is Empty\n";
1032         system("pause");
1033     }
1034     else
1035     {
1036         cout << "\nDelete: " << static_arr[h++] << "\n";
1037         if (h > t) // Reset the queue if all elements are popped
1038         {
1039             h = t = -1;
1040         }
1041         system("pause");
1042     }
1043     c--;
1044 }
1045
1046 int Static_Liner_Queue_isfull()
1047 {
1048     if (t == sizes - 1)
1049         return 1;
1050     else
1051         return 0;
1052 }
1053
1054 int Static_Liner_Queue_isempty()
1055 {
1056     return (h == -1 || h > t) ? 1 : 0;
1057 }
1058
1059 void queue_print()
1060 {
1061
1062     if (Static_Liner_Queue_isempty())
1063     {
1064         cout << "        Queue Is Empty\n\n";
1065     }
1066     else
1067     {
1068         for (int i = h; i <= t; i++)
1069         {
1070             cout << "    " << static_arr[i];
1071         }
1072         cout << "\n\n";
1073     }
1074 }
1075 //*****
1076
1077 void my_Static_Circular_Queue()

```

```

1078 {
1079     system("cls");
1080     int op, val;
1081     cout << " ---Circular Queue size Is "<<size<<"---\n\n";
1082     Circular_queue_print();
1083     cout << "*****\n";
1084     cout << "The Operation Of Circular Queue\n\n";
1085     cout << " 1->Push\n";
1086     cout << " 2->Pop\n\n";
1087     cout << "Press 0 to go back\n\n";
1088     cout << "Insert The Operation Number-> ";
1089     cin >> op;
1090
1091     switch (op)
1092     {
1093         case 1:
1094             if (Static_Circular_Queue_isfull() == 0)
1095             {
1096                 cout << "\nInsert The Value : ";
1097                 cin >> val;
1098                 Static_Circular_Queue_Push(val);
1099                 cout << "\n";
1100             }
1101             else
1102             {
1103                 cout << "\n";
1104                 cout << "Queue Is Full\n";
1105                 cout << "\n\n";
1106                 system("pause");
1107             }
1108
1109             my_Static_Circular_Queue();
1110             break;
1111
1112         case 2:
1113             if (Static_Circular_Queue_isempty() == 0)
1114                 Static_Circular_Queue_Pop();
1115             else
1116             {
1117                 cout << "\n";
1118                 cout << "Queue Is Empty\n";
1119                 cout << "\n\n";
1120                 system("pause");
1121             }
1122             my_Static_Circular_Queue();
1123             break;
1124
1125         case 0:
1126             h = -1;
1127             t = -1;
1128             c = 0;
1129
1130             staticMenu();
1131             break;
1132         default:
1133             cout << "\n";
1134             cout << "Incorrect Operation\n\n";
1135             cout << "\n\n";
1136             system("pause");
1137             my_Static_Circular_Queue();

```

```

1138         break;
1139     }
1140 }
1141
1142 void Static_Circular_Queue_Push(int val)
1143 {
1144     if (Static_Circular_Queue_isfull())
1145     {
1146         cout << "Queue is full, cannot push value.\n";
1147         return;
1148     }
1149
1150     if (h == -1)
1151         h = 0;
1152
1153     t = (t + 1) % sizes;
1154     static_arr[t] = val;
1155     c++;
1156 }
1157
1158 void Static_Circular_Queue_Pop()
1159 {
1160     if (Static_Circular_Queue_isempty())
1161     {
1162         cout << "Queue is empty, cannot pop value.\n";
1163         return;
1164     }
1165
1166     cout << "Delete: " << static_arr[h] << "\n";
1167     h = (h + 1) % sizes;
1168     c--;
1169
1170     if (c == 0)
1171     {
1172         h = -1;
1173         t = -1;
1174     }
1175
1176     system("pause");
1177 }
1178
1179 int Static_Circular_Queue_isempty()
1180 {
1181     return c == 0;
1182 }
1183
1184 int Static_Circular_Queue_isfull()
1185 {
1186     return c == sizes;
1187 }
1188
1189 void Circular_queue_print()
1190 {
1191     if (Static_Circular_Queue_isempty())
1192     {
1193         cout << "Queue Is Empty\n\n";
1194     }
1195     else
1196     {
1197         int i = h;

```

```

1198         for (int count = 0; count < c; count++)
1199         {
1200             cout << " " << static_arr[i];
1201             i = (i + 1) % sizes;
1202         }
1203         cout << "\n\n";
1204     }
1205 }
1206
1207 //*****
1208
1209 int main()
1210 {
1211     system("cls");
1212     int select;
1213     cout << "\n";
1214     cout << "          ----- \n";
1215     cout << "          |          (Choose)          | \n";
1216     cout << "          ----- \n";
1217     cout << "          | 1: Dynamic    2: Static    | \n";
1218     cout << "          ----- \n\n";
1219     cout << "Press 0 To Stop the program\n\n";
1220     cout << "Enter The Number -> ";
1221     cin>>select;
1222     switch(select)
1223     {
1224         case 1:
1225             dynamicMenu();
1226             break;
1227         case 2:
1228             staticMenu();
1229             break;
1230         case 0:
1231             system("cls");
1232             return 0;
1233             break;
1234         default:
1235             main();
1236             break;
1237     }
1238
1239
1240
1241 }
1242

```