الاسم :محمد علي رجب علي

مجموعـة : 2

سيكشـن : 4

University card

ASSIGNMENT

Data structures

Code Implementation

- (linked stack)
- (linked queue)
- (Selection Sort)
- (Insertion Sort)

Application:

- Converting Infix to Postfix Notation

Linked stack With Implementation

الاسم :محمد علي رجب علي

مجموعة : 2

سيكشن : 4

University card

```
#include<iostream>
using namespace std;
template<class t>
class Stack {
private:
 struct StackNode
 {
  t item;
  StackNode*next;
 };
 StackNode *topPtr, *curPtr;
public:
 Stack() {
  topPtr = NULL;
 }
 bool isEmpty()
 {
  return topPtr == NULL;
 }
 void push(t newItem)
 {
  StackNode *newPtr = new StackNode;
  if (newPtr == NULL)
   cout << "Stack push cannot allocate memory";
  else
  {
   newPtr->item = newItem;
   newPtr->next = topPtr;
   topPtr = newPtr;
  }
 }
 void pop() {
  if (isEmpty())
   cout << "Stack empty on pop";
  else {
   StackNode *temp = topPtr;
   topPtr = topPtr->next;
   temp->next = NULL;
   delete temp;
  }
 }
 void pop(t stackTop)
 {
  if (isEmpty())
   cout << "Stack empty on pop";
  else {
   stackTop = topPtr->item;
   StackNode *temp = topPtr;
   topPtr = topPtr->next;
   temp->next = NULL;
   delete temp;
  }
 }
 void getTop(t stackTop)
 {
  if (isEmpty())
   cout << "stack empty on getTop";
  else
  stackTop = topPtr->item;
  cout << "\nTop Element of the stack is " << stackTop;
  cout << endl;
 }
 void display()
 {
  curPtr = topPtr;
  cout << "\nItems in the stack : ";
  cout << "[ ";
  while (curPtr != NULL)
  {
   cout <<curPtr->item;
   curPtr = curPtr->next;
  }
  cout << " ]\n";
 }
};


int main()
{
 Stack<int>s;
 s.push(10);
s.push(20);
s.push(40);
s.push(60);
s.push(70);
 s.display(); // before  pop();
 s.pop(70)
s.push(80);
 s.display();
```

Output

before pop()

```
[70,60,40,20,10]
```

after pop()

```
[80,60,40,20,10]
```

Linked Queue With Implementation

الاسم :محمد علي رجب علي

مجموعة : 2

سيكشن : 4

University card

```
#include <iostream>
#include <cassert>
using namespace std;
template<class t>
class linkedQueue
{
private:
 struct Node
 {
  t item;
  Node *next;
 };
 int length;
 Node *frontPtr, *rearPtr;

public:

linkedQueue():frontPtr(NULL), rearPtr(NULL), length(0)
 {}
 bool isEmpty()
 {
  return (length == 0);
 }

 void dequeue()
 {
  if (isEmpty())
   cout << "Empty Queue" << endl;
  else if (length == 1)
  {
   delete frontPtr;
   rearPtr = NULL;
   length = 0;
  }
  else
  {
   Node *current = frontPtr;
   frontPtr = frontPtr->next;
   delete current;//free(current)
   length--;
  }
 }

 void enqueue(t item)
 {
  Node *newNode = new Node;
  newNode->item = item;
  newNode->next = NULL;

  if (length == 0)
   rearPtr = frontPtr = newNode;
  else
  {
   rearPtr->next = newNode;
   rearPtr = newNode;
  }
  length++;
 }

 t front()
 {
  assert(!isEmpty());
  return frontPtr->item;
 }

 t rear()
 {
  assert(!isEmpty());
  return rearPtr->item;
 }

 void clearQueue()
 {
  Node *current;

  while (frontPtr != NULL)
  {
   current = frontPtr;
   frontPtr = frontPtr->next;
   delete current;
  }
  rearPtr = NULL;
  length = 0;
 }
 void display()
 {
  Node*cur = frontPtr;
  cout << "Item in the queue :[ ";
  while (cur!=NULL)
  {
   cout << cur->item<<" ";
   cur = cur->next;
  }cout << "]" << endl;
 }

 void search(t item)
 {
  Node*cur = frontPtr;
  bool flag = true;
  while (cur != NULL)
  {
   if (cur->item == item)
   {
    cout << "the item :" << item << " found" << endl;
    flag = false;
    break;
   }
   cur = cur->next;
  }
  if(flag)
   cout << "the item : " << item << " not found" << endl;

 }

};

int main()
{
 linkedQueue<int>q1;

 for (int i = 1; i <= 20; i++)
  q1.enqueue(i);

 cout << q1.front() << endl;
 cout << q1.rear() << endl;
 q1.display();
 return 0;
}
```

Output

```
1
20
Item in the queue :[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ]
```

Insertion Sort With Implementation

الاسم :محمد علي رجب علي

مجموعة : 2

سيكشن : 4

University card

```
#include<iostream>
using namespace std;
void insertionSort(int arr[], int n)
{
 int key, j;
 for (int i = 1; i < n; i++)
 {
  key = arr[i];
  j = i - 1;

  while (j >= 0 && arr[j] < key)
  {
   arr[j + 1] = arr[j];
   j = j - 1;
  }
  arr[j + 1] = key;
 }
}
void printArray(int arr[], int n)
{
 for (int i = 0; i < n; i++)
  cout << arr[i] << " ";
 cout << endl;
}


/*
void insertionSortRecursive(int arr[], int n)
{
 // base case
 if (n <= 1)
  return;

 insertionSortRecursive(arr, n - 1);
 int last = arr[n - 1];
 int j = n - 2;

 while (j >= 0 && arr[j] > last)
 {
  arr[j + 1] = arr[j];
  j--;
 }
 arr[j + 1] = last;
}
*/
int main()
{

  int arr[] = { 80 , 90 ,60 ,30 ,50 ,70 ,40 };
  int n = sizeof(arr) / sizeof(arr[0]);//28/4=7

  insertionSort(arr, n);
  //insertionSortRecursive(arr, n);
  printArray(arr, n);

 return 0;

}
```

Output

```
90, 80, 70 ,60, 50, 40 ,30
```

Selection Sort With Implementation

الاسم :محمد علي رجب علي

مجموعة : 2

سيكشن : 4

University card

```
#include <iostream>
#include<algorithm>
using namespace std;
void selectionSort(int arr[], int n)
{
 int minIdx;
                      //0  1  2  3  4  5
 for (int i = 0; i < n - 1; i++)//60 40 50 30 10 20
 {
  minIdx = i;//4

  for (int j = i + 1; j < n; j++)
   if (arr[j] > arr[minIdx])
    minIdx = j;
         swap(arr[minIdx], arr[i]);

 }
}
/*
//\\//\\//\\//\\//\\Recursive//\\//\\//\\//\\//\\//\\
int minIndex(int a[], int i, int j)
{
 if (i == j)
  return i;

 int k = minIndex(a, i + 1, j);

 return (a[i] < a[k]) ? i : k;
}


void recurSelectionSort(int a[], int n, int index = 0)
{

 if (index == n)
  return;

 int k = minIndex(a, index, n - 1);

 if (k != index)
  swap(a[k], a[index]);

 recurSelectionSort(a, n, index + 1);
}

*/


void print(int arr[], int size)
{

 for (int i = 0; i < size; i++)
  cout << arr[i] << " ";
 cout << endl;

}

int main()
{
 int arr[] = { -60, 0, 50, 30, 10,20 };
 int n = sizeof(arr) / sizeof(arr[0]);//6*4=24  /  4
 selectionSort(arr, n);
  //recurSelectionSort(arr, n);
 cout<<"Array After Selection Sort : \n";
 print(arr, n);

 return 0;
 }
```

Output

```
Array After Selection Sort :
[50 ,30 20 ,,10 ,0 ,-60]
```

Application: Converting Infix to Postfix Notation

الاسم :محمد علي رجب علي

مجموعـة : 2

سيكشـن : 4

University card

```
...
#include <tqueue.h>
#include <tstack.h>
...
typedef fsu::TQueue < Token > TokenQueue;
typedef fsu::TStack < Token > TokenStack;
// a Token is either an operand, an operator, or a left or right parenthesis
...
bool i2p (TokenQueue & Q1, TokenQueue & Q2)
// converts infix expression in Q1 to postfix expression in Q2
// returns true on success, false if syntax error is encountered
{
  Token L( '(' ), R( ')' );  // left and right parentheses
  TokenStack S;              // algorithm control stack
  Q2.Clear();               // make sure ouput queue is empty
  while (!Q1.Empty())
  {
    if (Q1.Front() == L) // next Token is '('
    // push '(' to mark beginning of a parenthesized expression
    {
      S.Push(Q1.Front());
      Q1.Pop();
    }
    else if (Q1.Front().IsOperator()) // next Token is an operator
    {
      // pop previous operators of equal or higher precedence to output
      while (!S.Empty() && S.Top() >= Q1.Front())
      {
        Q2.Push(S.Top());
        S.Pop();
      }
      // then push new operator onto stack
      S.Push(Q1.Front());
      Q1.Pop();
    }
    else if (Q1.Front() == R) // next Token is ')'
    // regurgitate operators for the parenthesized expression
    {
      while (!S.Empty() && !(S.Top() == L))
      {
        Q2.Push(S.Top());
        S.Pop();
      }
      if (S.Empty())     // unbalanced parentheses
      {
        std::cout << "** error: too many right parens\n";
        return false;
      }
      S.Pop();          // discard '('
      Q1.Pop();          // discard ')'
    }
    else                 // next Token should be an operand
    // send operand directly to output
    {
      Q2.Push(Q1.Front());
      Q1.Pop();
    }
  } // end while()
  // regurgitate remaining operators
  while (!S.Empty())
  {
    if (S.Top() == L)      // unbalanced parentheses
    {
      std::cout << "** error: too many left parens\n";
      return false;
    }
    Q2.Push(S.Top());
    S.Pop();
  }
  return true;
} // end i2p()
```