Search Algorithms

* Linear Search
* *#include*<iostream>
* using namespace std;
* *//linear search*
* int list[50]; *//initialize an array called list with a size of 50*
* int n, i; *//initialize integers n and i*
* void captureinputs()
* {
* cout<<"Please enter the size of values you want to work with; ";
* cin>>n;
* cout<<"Please enter the values; ";
* *for*(i=0; i<n; i++)
* {
* cin>>list[i]; *//capture the values of the array after specifying the size of the array*
* }
* }
* void linearsearch()
* {
* int key;
* cout<<"Please enter the value you are looking for: ";
* cin>>key; *//enter the value you want to look for in the array*
* *for*(i=0; i<n; i++)
* {
* *if*(key==list[i]) *//comparing the key to the value of the array at the start of the array as it loops through the for loop*
* {
* cout<<"key"<<" has been found at location "<<i; *//display that it has been found if the key is equal to a value in the list*
* *break*; *//meant to stop the search if it has not been found*
* }
* }
* *if*(i>=n) *//if we are going past the size of the array and the key has not been found*
* {
* cout<<key<<" was never found.:("; *//display key was never found*
* }
* }
* int main()
* {
* captureinputs();
* *linearsearch();*
* }
* Binary Search
* *#include*<iostream>
* using namespace std;
* *//linear search*
* int list[50]; *//initialize an array called list with a size of 50*
* int n, i; *//initialize integers n and i*
* void captureinputs()
* {
* cout<<"Please enter the size of values you want to work with; ";
* cin>>n;
* cout<<"Please enter the values; ";
* *for*(i=0; i<n; i++)
* {
* cin>>list[i]; *//capture the values of the array after specifying the size of the array*
* }
* }
* void binarysearch()
* {
* int key;
* cout<<"Please enter the value you are looking for: ";
* cin>>key; *// capture the value you want to search for in the array*
* int low = 0; *//initializing low to be zero, the start of the array*
* int high = n-1; *//initializing high to be n-1, the highest value in the array*
* int mid; *//creating variable mid*
* *for*(i=0; i<n;i++)
* {
* mid=(low+high)/2; *//defining what mid equals to*
* *if*(key==list[mid]) *//comparing the keyvalue to the values in the array*
* {
* cout<<key<<" was found at location "<<mid; *//display the key value and the index it was found in if found*
* *break*; *//break if found or not*
* }
* *else* *if*(key<list[mid]) *//if not found and if key is less than the value at the mid point, change the high point to high point minus one*
* {
* high=high-1;
* }
* *else*
* {
* low=low+1; *//if not found and if key is more than the value at the mid point, change the low point to low point plus one*
* }
* }
* *if*(low>high)
* {
* cout<<key<<" was never found."; *//printed if not found at all*
* }
* }
* int main()
* {
* captureinputs();
* binarysearch();
* }

Sort Algorithms

* Selection Sort
* *#include*<iostream>
* using namespace std;
* *//selection sort*
* *//has unsorted values*
* int i, j, n;
* int list[10];
* void captureinputs()
* {
* cout<<"Please enter the number of values you want to capture: ";
* cin>>n;
* cout<<"Please enter the values: ";
* *for*(i=0; i<n;i++)
* {
* cin>>list[i];
* }
* }
* void display()
* {
* *for*(i=0; i<n; i++)
* {
* cout<<list[i]<<" ";
* }
* }
* void selectionsort()
* {
* int min;
* *for*(i=0; i<n; i++)
* {
* min=i;
* *for*(j=i+1; j<n; j++)
* {
* *if*(list[j]<list[min])
* {
* min=j;
* }
* }
* swap(list[i], list[min]);
* }
* }
* int main()
* {
* captureinputs();
* selectionsort();
* display();
* }
* Insertion Sort
* *#include*<iostream>
* using namespace std;
* *//selection sort*
* *//has unsorted values*
* int i, j, n;
* int list[10];
* void captureinputs()
* {
* cout<<"Please enter the number of values you want to capture: ";
* cin>>n;
* cout<<"Please enter the values: ";
* *for*(i=0; i<n;i++)
* {
* cin>>list[i];
* }
* }
* void display()
* {
* *for*(i=0; i<n; i++)
* {
* cout<<list[i]<<" ";
* }
* }
* *//insertion sort*
* *void insertionsort()*
* *{*
* *for(i=1; i<n; i++)*
* *{*
* *j=1;*
* *while(list[j]<list[j-1])*
* *{*
* *//swap(list[j], list[j-1]);*
* *int temp = list[j];*
* *list[j] = list[j-1];*
* *list[j-1]=temp;*
* *j--;*
* *}*
* *}*
* *}*
* int main()
* {
* captureinputs();
* *insertionsort();*
* display();
* }
* Bubble Sort

void bubbleSort(){

for (i=0 ; I<n ; I++){

for (j =0; j<n-1 ; j++){

if (list[j] > list[j+1){

swap(list[j], list[j+1]

}

}

}

}

Storage Data Structures

* Arrays
* *#include*<iostream>
* using namespace std;
* int main() {
* *int list[50];*
* *int n, i;*
* *cout<<"How many values do you want to work with? ";*
* *cin>>n;*
* *cout<<"Enter values now: ";*
* *for (i=0; i<n; i++)*
* *{*
* *cin>>list[i];*
* *}*
* *cout<<"Values are "<<endl;*
* *for (i=0; i<n; i++)*
* *{*
* *cout<<list[i]<<"\t";*
* *}*
* }
* Stacks
* *#include*<iostream>
* *#define* size 60
* *//stack structure //goes last in first out*
* using namespace std;
* class stack{*//creating stack structure to hold arrays*
* int top;
* int list[size];
* public:
* stack();
* void push(int item);*//insert item*
* int pop();*//remove*
* int peek();*//see*
* void display();*//display*
* };
* stack::stack()
* {
* top=-1;
* }
* void stack::push(int item)
* {
* *if*(top==size)
* {
* cout<<" stack overflow ";*//if full*
* *return*;
* }
* *else*
* {
* top=top+1;
* list[top]=item;
* }
* }
* int stack::pop()*//removing*
* {
* *if*(top==-1)
* {
* cout<<" stack underflow ";
* *return* 0;
* }
* *else*
* {
* int item=list[top];
* top--;
* *return* item;
* }
* }
* int stack::peek()*//trying to see what is there*
* {
* *if*(top==-1)
* {
* cout<<" stack underflow ";
* }
* *else*
* {
* int item=list[top];
* top--;
* *return* item;
* }
* }
* void stack::display()
* {
* *while*(top!=-1)
* {
* cout<<pop()<<" ";
* *//cout<<list[top]<<" "*
* *//top--*
* }
* }
* int main()
* {
* stack s;
* int i, n;
* int item;
* cout<<" how many values ";
* cin>>n;
* cout<<" enter values";
* *for*(i=0; i<n; i++)
* {
* cin>>item;
* s.push(item);
* }
* cout<<" item at the top is "<<s.peek();
* cout<<" \n items are ";
* s.display();
* }

Stack with Linked List

*#include*<iostream>

using namespace std;

struct node{

    int data;

    node \*next;

};

class stack{

    node \*head;

    public:

        stack();

        void push(int item);

        void display();

};

stack::stack()

{

    head=NULL;

}

void stack::push(int item)

{

    node \*newnode=new node;

    newnode->data=item;

    newnode->next=NULL;

*if*(head==NULL)

    {

        head=newnode;

    }

*else*

    {

        newnode->next=head;

        head=newnode;

    }

}

void stack::display()

{

    node \*temp;

    temp=head;

*while*(temp!=NULL)

    {

        cout<<temp->data<<" ";

        temp=temp->next;

    }

}

int main()

{

    stack s;

    int i, n;

    int item;

    cout<<"how many values ";

    cin>>n;

    cout<<"enter values ";

*for* (i=0; i<n; i++)

    {

        cin>>item;

        s.push(item);

    }

    s.display();

}

Stack assignment

*#include* <iostream>

using namespace std;

*#define* size 7

class stack

{

    int top;

    string arr[size];

    public: stack(); *//constructor*

    public: void push(string item);

    public: string pop();

    public: string peek();

    public: void display();

};

stack::stack()

{

    top = -1;

}

void stack::push(string item)

{

*if* (top == size-1)

    {

        cout << "Stackoverflow";

*return*;

    }

*else*

    {

        top= top+1;

        arr[top]= item ;

    }

}

string stack::pop()

{

*if* (top == -1)

    {

        cout << "stackunderflow";

*return* 0;

    }

*else*

    {

        string item = arr[top];

        top--;

*return* item;

    }

}

string stack::peek()

{

*if* (top == -1)

    {

        cout << "stackunderflow";

*return* 0;

    }

*else*

    {

        string item = arr[top];

*return* item;

    }

}

void stack::display()

{

*while* (top != -1)

    {

        cout << pop() << " ";

    }

}

int main()

{

    stack s;

    int len, i;

    string name;

    cout << "Please enter the number of students in your class: ";

    cin >> len;

    cout << "enter names: \n";

*for* (i = 0; i < len; i++) {

        cin >> name;

        s.push(name);

    }

    cout << "\nThe name at the top of the stack is: " << s.peek();

    cout << "\nThe names of the students are: ";

    s.display();

}

Stack Structure 2

*//stackStructure*

*#include*<iostream>

*#define* size 3

using namespace std;

class stack{

    int top; *//pointer*

    int list[size];

    public:

stack();

void push(int item);

int pop();

int peek();

void display();

};

stack::stack() *//constructor*

{

    top=-1;

}

void stack::push(int item)

{

*if* (top==size-1)

    {

        cout<<"stack overflow\n ";

*return*;

    }

*else*

    {

        top=top+1;

        list[top]=item;

    }

}

int stack::pop()

{

*if*(top==-1)

    {

        cout<<"\nstack underflow\n";

*return* 0;

    }

*else*

    {

        int item=list[top];

        top--;

*return* item;

    }

}

int stack::peek()

{

*if*(top==-1)

    {

        cout<<"\nstack underflow\n";

    }

*else*

    {

        int item=list[top];

*return* item;

    }

}

void stack::display()

{

*while*(top!=-1)

    {

        cout<<pop()<<" ";

    }

}

int main()

{

    stack s;

    int i, n;

    int x;

    int item;

    cout<<" how many values ";

    cin>>n;

    cout<<"enter values ";

*for*(i=0; i<n;i++)

    {

        cin>>item;

        s.push(item);

    }

    cout<<" item at the top is "<<s.peek();

    cout<<" items are "<<"\n";

    s.display();

}

* Queues
* *//queues*
* *#include*<iostream>
* *#define* size 40
* using namespace std;
* class queue{
* int rear, front;
* int list[size];
* public:
* queue();
* void enqueue(int item);
* int dequeue();
* void display();
* };
* queue::queue()
* {
* rear=-1;
* front=-1;
* }
* void queue::enqueue(int item)
* {
* *if*(rear==size-1)
* {
* cout<<"Queue Overflow";
* }
* *else* *if* (rear==-1)
* {
* rear++;
* list[rear]=item;
* front++;
* }
* *else*
* {
* rear++;
* list[rear]=item;
* }
* }
* int queue::dequeue()
* {
* *if*(front==-1)
* {
* cout<<"Queue Underflow";
* *return* 0;
* }
* *else* *if*(rear==front)
* {
* int item=list[front];
* rear=front=-1;
* *return* item;
* }
* *else*
* {
* int item=list[front];
* front++;
* *return* item;
* }
* }
* void queue::display()
* {
* *for* (int i=front; i<=rear;i++)
* {
* cout<<dequeue()<<" ";
* }
* }
* int main()
* {
* queue q;
* int i, n, item;
* cout<<"�nter the number of values";
* cin>>n;
* cout<<"Enter values: ";
* cin>>item;
* *for* (i=0;i<n;i++)
* {
* cin>>item;
* }
* q.display();
* }

Queue Assignment

*#include* <iostream>

using namespace std;

*#define* size 7

class queue{

 int rear, front;

 string arr[size];

 public:

  queue();

  void enqueue(string item);

        string dequeue();

        string peek();

        void display();

};

queue::queue(){

     rear = -1;

     front = -1;

 }

void queue::enqueue(string item)

{

*if* (rear==size-1)

 {

  cout<<"Queue overflow";

*return*;

 }

*else* *if* (rear == -1)

 {

  rear++;

  arr[rear]= item;

  front++;

 }

*else*

 {

  rear++;

  arr[rear]=item;

 }

}

string queue::dequeue()

{

*if* (front==-1)

 {

  cout<<"queueunderflow";

*return* 0;

 }

*else* *if* (rear==front)

 {

  string item = arr[front];

  rear = front = -1;

*return* item;

 }

*else*

 {

  string item = arr[front];

  front++;

*return* item;

 }

}

string queue::peek()

{

*if* (front==-1)

 {

  cout<<"queueunderflow";

*return* 0;

 }

*else*

 {

  string item = arr[front];

*return* item;

 }

}

void queue::display()

{

*for* (int i = front; i<=rear; i++){

  cout<<dequeue()<<" ";

 }

}

int main(){

 queue q;

 int len, i;

 string name;

 cout<<"Please enter the number of students in your class: ";

 cin>> len;

 cout<<"Please enter the student(s) name(s): \n";

*for* (i=0; i<len; i++){

  cin>>name;

  q.enqueue(name);

 }

 cout<<"\nThe name at the front of the queue is: "<<q.peek();

 cout<<endl;

 cout<<"\nThe list of students is: ";

 q.display();

}

* Linked List

Link Each link of a linked list can store a data called an element.

Next Each link of a linked list contains a link to the next link called Next.

Linked List A Linked List contains the connection link to the first link called First.

* #include <iostream>
* using namespace std;
* struct Node {
* int data;
* struct Node \*next;
* };
* struct Node\* head = NULL;
* void insert(int new\_data) {
* struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));
* new\_node->data = new\_data;
* new\_node->next = head;
* head = new\_node;
* }
* void display() {
* struct Node\* ptr;
* ptr = head;
* while (ptr != NULL) {
* cout<< ptr->data <<" ";
* ptr = ptr->next;
* }
* }
* int main() {
* insert(3);
* insert(1);
* insert(7);
* insert(2);
* insert(9);
* cout<<"The linked list is: ";
* display();
* return 0;
* }