1. **Wildcard pattern Matching O(2^(number of character in string))**

#include<bits/stdc++.h>  
using namespace std;  
bool match(stack<char> p,stack<char> s){  
 bool res = false;  
 if(p.size()==0 && s.size()==0)return true;  
 else if(p.size()==0 && s.size()>0)return false;  
  
 else if(p.top()==s.top()){  
 p.pop();s.pop();  
 return match(p,s);  
 }  
 else if(p.top()=='?'){  
 p.pop();s.pop();  
 return match(p,s);  
 }  
 else if(p.top()=='\*'){  
 p.pop();  
 if(p.size()==0)return true;  
 while(!res && s.size()>0){  
 res |= match(p,s);  
 s.pop();  
 if(res)return res;  
 }  
 }  
 return false;  
}  
int main()  
{  
 string wild ="z\*ho",orig = "zoho";  
 stack<char> p,s;  
 for(int i=0;i<wild.size();i++){  
 p.push(wild[i]);  
 }  
 for(int i=0;i<orig.size();i++){  
 s.push(orig[i]);  
 }  
 cout<<match(p,s);  
 return 0;  
}

|  |  |
| --- | --- |
| **Vector** | **List** |
| * It has contiguous array. * It is synchronized, so it is thread safe. * Insertion in end almost constant time in maximum cases but insertion in middle is always costly. * Doesn’t need additional memory. * Random access of elements is possible. | * It has non contiguous array. * It is not synchronized , so it causes inconsistency in data. * Insertion is constant time at any location. * Needs additional memory to hold previous and next data nodes. * Random access of elements is not possible. |

**2.)Difference between vector,list,map and hash\_map**

Appliations of vector and list:

* Implementation of stacks and queues
* Implementation of graphs : Adjacency list representation of graphs

|  |  |
| --- | --- |
| **Map** | **HashMap/UnOrderedMap** |
| * Requires O(log(N) )time for inserts and find operations. * It is implemented using **Red-Black Tree** data structure. * It is slow. | * Requires an average time of O(1) for inserts and find. * It is implemented using **Hash Table** data structure. * It is synchronized, So it is thread safe. * It is fast. |

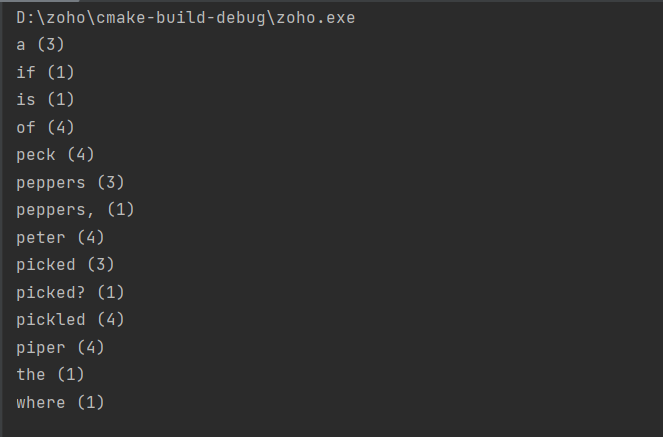
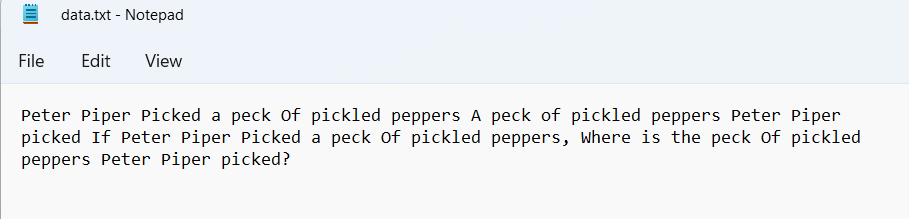
Applications of map and hashamp:

* when we need efficient implementation of search, insert and delete operations.
* DNS Table
* ARP Table

**3.)Using List : Design an Application that reads a Document from a file and outputs the distinct words and their frequencies in alphabetical order.**

**O(NlogN)**

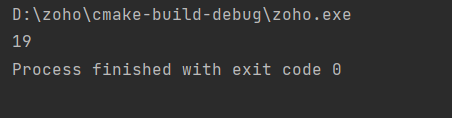
#include<bits/stdc++.h>  
using namespace std;  
class wordF{  
 int freq=0;  
 public:  
 string word;  
 int getFreq(){return freq;}  
 void incFreq(){freq++;}  
 wordF(string word){  
 this->word = word;  
 }  
};  
  
int main()  
{  
  
 string word;  
 ifstream file("D://data.txt");  
 list<wordF> dataFreq;  
  
 while (file>>word) {  
 bool present = false;  
 transform(word.begin(), word.end(), word.begin(), ::tolower);  
 for(wordF &i : dataFreq){  
 if(i.word==word){  
 i.incFreq();  
 present = true;  
 break;  
 }  
 }  
  
 if(!present){  
 wordF w(word);  
 w.incFreq();  
 dataFreq.push\_back(w);  
 }  
 }  
 file.close();  
 dataFreq.sort([](const wordF &a,const wordF &b){  
 return a.word<b.word;  
 });  
  
 for(wordF i : dataFreq){  
 cout<<i.word<<" ("<<i.getFreq()<<")\n";  
 }  
 return 0;  
}

**** ****

**4.)Expression evaluation**

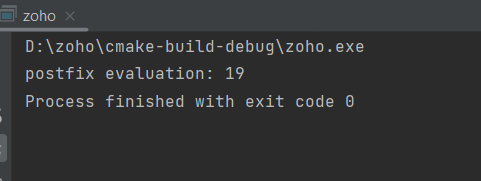
**Infix evaluation**

#include<bits/stdc++.h>  
using namespace std;  
  
unordered\_map<char,int> precd;  
  
void doProcess(stack<int> &nums,stack<char> &opr){  
  
 int a = nums.top();nums.pop();  
 int b = nums.top();nums.pop();  
 if(opr.top()=='+'){  
 nums.push(a+b);  
 }  
 else if(opr.top()=='-'){  
 nums.push(b-a);  
 }  
 else if(opr.top()=='\*'){  
 nums.push(a\*b);  
 }  
 else{  
 if(a==0){  
 cout<<"Cannot divide by 0";  
 exit(0);  
 }  
 nums.push(b/a);  
 }  
 opr.pop();  
}  
  
int main()  
{  
 string s="( 2 + 5 ) \* 3 - 8 / 3";  
 precd = {{'\*',2},{'/',2},{'+',1},{'-',1}};  
 stack<int> nums;  
 stack<char> opr;  
 for(int i=0;i<s.size();i++){  
 if(s[i]==' ')continue;  
 if(isdigit(s[i])){  
 int n = 0;  
 while(i<s.size() && isdigit(s[i])){  
 n=n\*10+(s[i]-'0');  
 i++;  
 }  
 i--;  
 nums.push(n);  
 }  
 else if(s[i]=='('){  
 opr.push(s[i]);  
 }  
 else if(s[i]==')'){  
 while(opr.top()!='(')  
 doProcess(nums,opr);  
 opr.pop();  
 }  
 else{  
 while(!opr.empty() && precd[s[i]]<=precd[opr.top()]){  
 doProcess(nums,opr);  
 }  
 opr.push(s[i]);  
 }  
 }  
  
 while(!nums.empty() && !opr.empty()){  
 doProcess(nums,opr);  
 }  
 cout<<nums.top();  
 return 0;  
}

****

**Postfix evaluation O(N)**

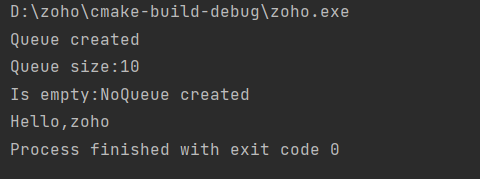
#include <bits/stdc++.h>  
  
using namespace std;  
  
int doProcess(string &exp)  
{  
 stack<char> stack;  
 int i;  
  
 for (i = 0; i<exp.size(); ++i)  
 {  
 if (isdigit(exp[i]))  
 stack.push(exp[i] - '0');  
 else  
 {  
 int val1 = stack.top();stack.pop();  
 int val2 = stack.top();stack.pop();  
 switch (exp[i])  
 {  
 case '+': stack.push(val2 + val1); break;  
 case '-': stack.push(val2 - val1); break;  
 case '\*': stack.push(val2 \* val1); break;  
 case '/': stack.push(val2/val1); break;  
 }  
 }  
 }  
 return stack.top();  
}  
  
int main()  
{  
 string s = "25+3\*83/-";  
 cout<<"postfix evaluation: "<< doProcess(s);  
 return 0;  
}



**5.) Custom data structure**

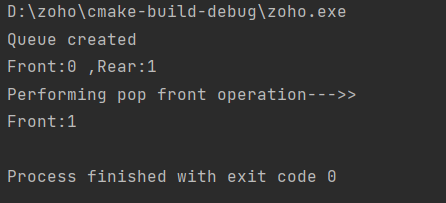
**Implementing Custom Queue:**

#include <bits/stdc++.h>  
  
using namespace std;  
template <typename T> class Queue{  
 vector<T> q;  
public:  
 Queue(){  
 cout<<"Queue created\n";  
 }  
 void push(T data){  
 q.push\_back(data);  
 }  
 bool isempty(){  
 return q.size()==0;  
 }  
 T top(){  
 if(isempty()){  
 return NULL;  
 }  
 return q[0];  
 }  
 int size(){  
 return q.size();  
 }  
 void pop(){  
 if(isempty())return;;  
 q.erase(q.begin());  
 }  
};  
int main()  
{  
 Queue<int> queue;  
 for(int i=0;i<10;i++)queue.push(i);  
 cout<<"Queue size:"<<queue.size()<<endl;  
 cout<<"Is empty:"<<(queue.isempty()?"Yes":"No");  
 Queue<string> queue1;  
 queue1.push("Hello");  
 queue1.push(",zoho");  
 while(!queue1.isempty()){  
 cout<<queue1.top();  
 queue1.pop();  
 }  
 return 0;  
}



**Implementing custom DeQueue**

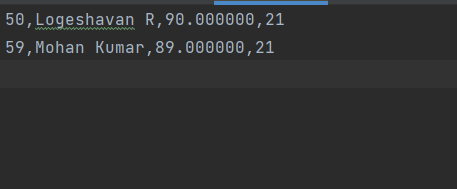
#include <bits/stdc++.h>  
  
using namespace std;  
template <typename T> class DeQueue{  
 list<T> q;  
public:  
 DeQueue(){  
 cout<<"Queue created\n";  
 }  
 void push\_back(T data){  
 q.push\_back(data);  
 }  
 void pop\_back(){  
 if(isempty())return;  
 q.pop\_back();  
 }  
 void push\_front(T data){  
 q.push\_front(data);  
 }  
 void pop\_front(){  
 if(isempty())return;  
 q.pop\_front();  
 }  
 bool isempty(){  
 return q.size()==0;  
 }  
 T front(){  
 if(isempty()){  
 return NULL;  
 }  
 return q.front();  
 }  
 T rear(){  
 if(isempty())return NULL;  
 return q.back();  
 }  
 int size(){  
 return q.size();  
 }  
};  
int main()  
{  
 DeQueue<int> queue;  
 queue.push\_back(1);  
 queue.push\_front(0);  
 cout<<"Front:"<<queue.front()<<" ,Rear:"<<queue.rear()<<endl;  
 queue.pop\_front();  
 cout<<"Performing pop front operation--->>\n";  
 cout<<"Front:"<<queue.front()<<endl;  
 return 0;  
}

****

**6. Implement the Job Interview Scheduling Application using queue. Assume if office opening time 9:00 AM closing time is 5:00 PM**

**7.write the student record in one file when we close our Application and while we execute the Application load the student record file into STL Container class hash\_map**

#include <bits/stdc++.h>  
using namespace std;  
  
struct Student {  
 string name;  
 int roll\_number;  
 int age;  
 double total\_marks;  
};  
unordered\_map<int,Student> students;  
void add(){  
 cout<<"=============>>>>>>>>>>Adding Student<<<<<<<<<<<<<<<=================\n";  
 Student student;  
 cout<<"Enter Student details::::\n\n";  
 cin.ignore();  
 cout<<"Name:";getline(cin,student.name);  
 cout<<"Age:";cin>>student.age;  
 cout<<"Roll Number:";cin>>student.roll\_number;  
 cout<<"Total Marks:";cin>>student.total\_marks;  
 cout<<"Student added successfully\n";  
 students[student.roll\_number] = student;  
}  
void onLoad(){  
 ifstream file;  
 file.open("data.txt",ios::in);  
 if (file.is\_open()){  
 string line;  
 while(getline(file, line)){  
 if(line=="")break;  
 int start = 0;  
 int end = line.find(",");  
 vector<string> values;  
 while (end != -1) {  
 values.push\_back(line.substr(start, end - start));  
 start = end + 1;  
 end = line.find(",", start);  
 }  
 values.push\_back(line.substr(start, end - start));  
 Student student;  
 student.roll\_number = atoi(values[0].c\_str());  
 student.name = values[1];  
 student.total\_marks = atoi(values[2].c\_str());  
 student.age = atoi(values[3].c\_str());  
 students[student.roll\_number] = student;  
 }  
 file.close();  
 }  
}  
void search(){  
 cout<<"=============>>>>>>>>>>Searching Student<<<<<<<<<<<<<<<=================\n";  
 cout<<"Student Roll Number:";  
 int roll;cin>>roll;  
 if(students.find(roll)== students.end())  
 cout<<"Student Not found with given roll number : "<<roll<<endl;  
 else{  
 Student value = students[roll];  
 cout<<"Student Details---:\n";  
 string t = ""+to\_string(value.roll\_number)+","+value.name+","+to\_string(value.total\_marks)+","+to\_string(value.age)+"\n";  
 cout << t;  
 }  
}  
void Delete(){  
 cout<<"=============>>>>>>>>>>Deleting Student<<<<<<<<<<<<<<<=================\n";  
 cout<<"Student Roll Number:";  
 int roll;cin>>roll;  
 if(students.find(roll)== students.end())  
 cout<<"Student Not found with given roll number : "<<roll<<endl;  
 else{  
 students.erase(roll);  
 cout<<"Deleted student\n";  
 }  
}  
void onExit(){  
 ofstream file;  
 file.open("data.txt",ios::out);  
 if (file.is\_open()){  
 for(const auto & [key,value] : students){  
 string t = ""+to\_string(value.roll\_number)+","+value.name+","+to\_string(value.total\_marks)+","+to\_string(value.age)+"\n";  
 file << t;  
 }  
 }  
 file.close();  
}  
void printStudents(){  
 cout<<"=================>>>>>>>>>>Student Records<<<<<<<<<<<<<<<<<<<==========================\n";  
 for(const auto & [key,value] : students){  
 string t = ""+to\_string(value.roll\_number)+","+value.name+","+to\_string(value.total\_marks)+","+to\_string(value.age)+"\n";  
 cout << t;  
 }  
}  
int main(){  
 onLoad();  
 cout<<"=================>>>>>>>>>>Student Record Management<<<<<<<<<<<<<<<<<<<==========================\n";  
 bool exit =false;  
 while(!exit){  
 cout<<endl;cout<<endl;  
 int choice;  
 cout<<"1.) Add Student\n2.) Search Student\n3.) Delete Student\n4.) print students\n5.) Exit\nyour choice:";  
 cin>>choice;  
 switch(choice){  
 case 1:  
 add();  
 break;  
 case 2:  
 search();  
 break;  
 case 3:  
 Delete();  
 break;  
 case 4:  
 printStudents();  
 break;  
 case 5:  
 exit = true;  
 break;  
 default:  
 cout<<"Wrong Choice------:(\n";  
 }  
 }  
 onExit();  
 return 0;  
}

****