CS-351 Data Structure Practical

Dated : 01:09.2024

Write a complete program to implement students Examination Records of an institute. Use doubly linked list for implementation you can use C/C++ Language. The attributes of record is as follows.

- Enrollement Number
- Student's Name
- Father's Name.
- Date of Birth
- Semester and year of admission
- Subject of study

The attribute of subject of a Semester

- Semester and year
- Subject code
- Subject title
- Maximum Mark for sessional
- Maximum Mark for theory
- Marks awarded in sessional
- Marks awarded in Theory

(A) The praggram should berform the following operations. Design an input panel to manage database system

- Add a scheme
- Delete a scheme
- Ubdate a scheme
- Print Rebort card
- Print report of results complete for asemester
- List record/result & provide provision to sort the list on

(B) (a) Number of Nodes bossible to create

(b) Time required to search a required

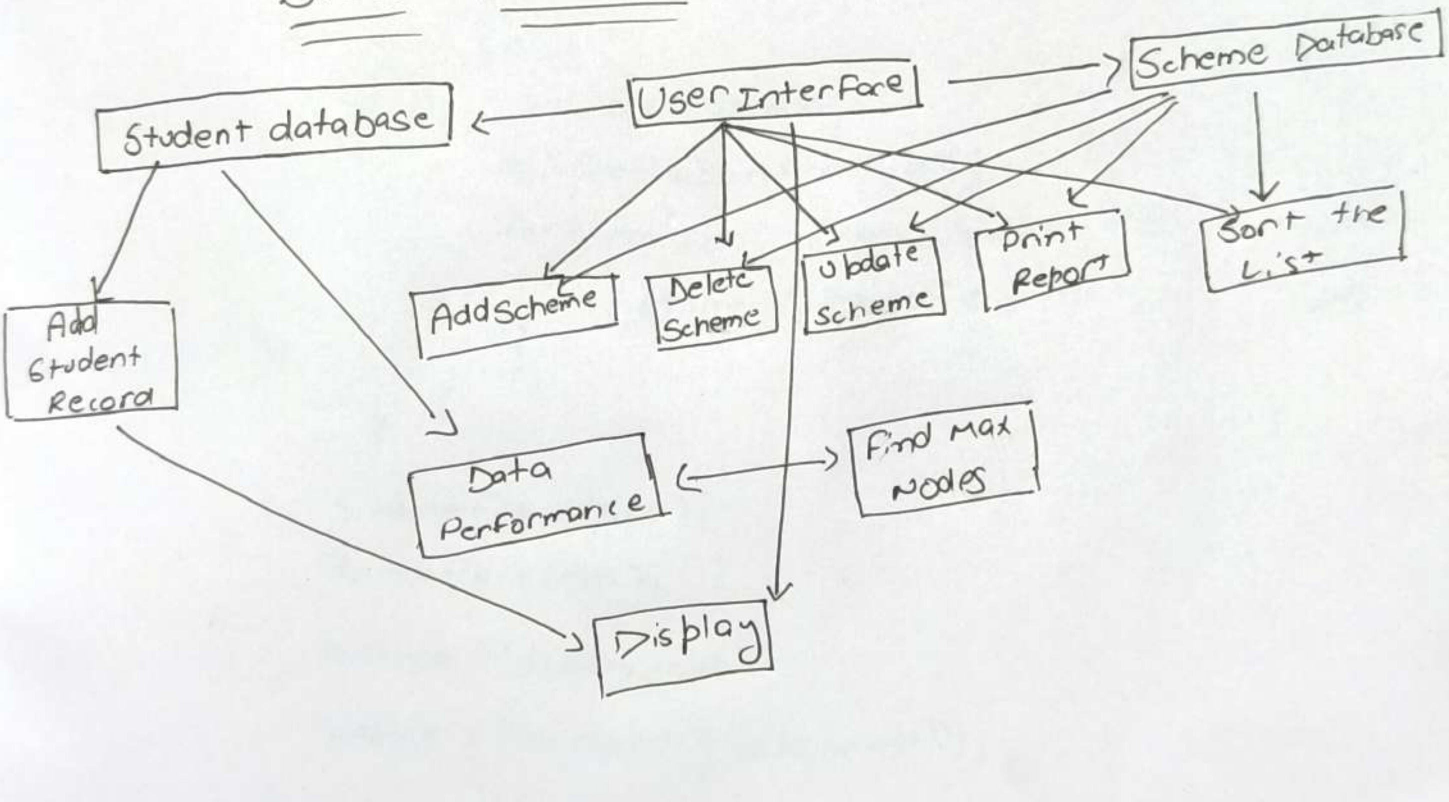
(c) Time required to dete a record

(d) Time required to insert a record

Discussion

The program uses a doubly linked list to manage student examination records, providing efficients data handling and manibulation. The design includes functionalities for managing Schemes, student records and performance metris while also handling file operation and user interactions. By following this model you can create a robust and efficient System for managing educational records.

Block DiAGRAM



Pseudocode to Delete Scheme

```
Void deleteScheme(){
    String schemeCode;
    O/P "Enter Scheme code to delete".
    I/P Scheme code;
    ifstream schemefile ("Scheme.txt");
    ofstream ~~scheme~~tempFile ("temp.txt');
    String line;
    while (getline (Schemefile, line)){
        IF(line.Find (Scheme Code)!=0){
            tempFile << line <<endl
        }else{
            int numSubjects;
            schemeFile>> numSubjects;
            for(int i=0; i <numSubjects: i++){
                getline (schemeFile, line);
            }
        }
    }
    Scheme.File.close();
    tempFile.close();
    remove ("scheme.txt");
    rename ("temp.txt", "Scheme.txt");
}
```

Pseudocode to update scheme

```
void updateScheme(){
        string schemeCode;
        o/p "Enter scheme code to update";
        I/p schemeCode;

    ifstream  Schemefile ("Scheme.txt");;
    ofstream  tempfile ("temp.txt");

    String line;
    While (getline(schemeFile, line)){
        IF(line.find(scheme Code)!=0){
            tempFile <<line<<Endl;
        } else {
            int numSubjects;
           Schemefile >> numSubjects;

           Scheme scheme;
          scheme.Scheme code ←schemeCode;
           o/p "Enter new scheme Name;
            cin.ignore;
           getline (cin, schem.schemeName);

           Subject* head =nullptr;
           subject* tail =nullptr;
       For(int i←0; i< numSubjects;i++){
           Subject* subject =new Subject;
             o/p " Enter new subject code;
           IP subject ->Subject code;
           o/p " Enter new subject title;
           getline (cin, subject ->subjecttitle);
             o/p "Enter newmarks"
           IP subject -> maxSessionalmarks
next[Subject] ← nullptr;

    IF (head = nullptr){
        head ← tail ← subject)
    }else{
    next[tail] ← subject;
        tail ← subject;
    }
  {
 Scheme.Subjects ← head;
 tempfile << Scheme.schemeCode << Scheme.schemeName << numSubjects

 subject* temp ← scheme.subjects;
 while (temp != nullptr){
 tempfile << temp->subjectCode << temp->subjectTitle << temp->maxses
 temp ← next[temp];
 }
 }
)
schemeFile.close();
temp File.close();
remove("scheme.txt");
rename ("temp.txt, "scheme.txt")
}
```

Pseudocode to print Report

```
void printReport ( StudentRecord * head ){
        String enrollmentNumber;
         O/P " Enter Number"
         I/P enrollement Number;
      StudentRecord* temb ← head;
      while (temp != nullptr){
        if ( temp [enrollment Number = enrollment Number){
        O/P "Report card " StudentName[temp]
        O/P "Enrollment No. " enrollmentNumber [temp]
        O/P " Father Name << FatherName[temp]
        O/P " DOB " << dateofBirth [temp]
        O/P "Semester & Year" << SemesterAndYear[temp]
        O/P "Scheme: << scheme[temp]

   Subject* tempSubjects ← Subjects [temp];
   while (tempSubject ← Subjects [temp] {
        O/P "Subject Code" subjectCode [temb Subject]
        O/P " Subject title" SubjectTitle[temp Subject]
        O/P " Maximum Sesh marks" maxSessionalmark [tempSubj
        O/P " Maximum Theory marks "
        O/P " Mark awarded in sesh "
        O/P " Mark awarded intheory"
        temb Subjects ← next [tenpSubject];

     }
    return;
  }
 temb ← next [temb;
  }
O/P "record not Found!
}
```

Pseudocode to Sort Record

```
void listandSort record(Student Record* head, bool byName ←true){
        studentRecord* arr[Size];
          int count ← 0;
        student Record* temb ← head;
  while(temb! = nullptr&& count< Size){
      arr[count++] ←temb;
        temb ← next[temb];
    }
  For (int i ← 0; i< count-1; i++){
     For (int j ← 0; i<count-i-1; j++){
        bool swab ← False;
      iF(byname){
        iF (studentName (arr[j]) > studentName (arr [j+1])
           swab ← true;
        }
    } else{
      IF(arr [j]enrollmentNumber) enrolmentNumber (arr[j+1]
           Swab ← true;
     }
   IF (Swab){
    Student Record* temb ← arr[j];
         arr[j] ← arr[j+1];
          arr [j+1] ← temb;
     }
   }
}
```

Pseudocode to measure Search time

```
void measureSearchTime (StudentRecord * head){
    if(head= nullptr){
        O/P"No record to search"
        return;
    }
auto start ← chrono::high_resolution_clock::now();
    StudentRecord *temb ← head;
    while (temp!=nullptr){
        IF (enrollmentNumber [temb] = SearchEnrollmentNumber{
            break;
        }
        temb ← next[temb];
    }
    auto end← chrona::high-resolution_clock::now();
    chrono:: duration<double> elapsed ← end-start;
    if (temb != nullptr){
        O/P "student record Found"
    } else
        O/P "Not Found"
    O/P "search time" elapsed.count() "seconds"
```

PseudoCode to Measure Max Nodes

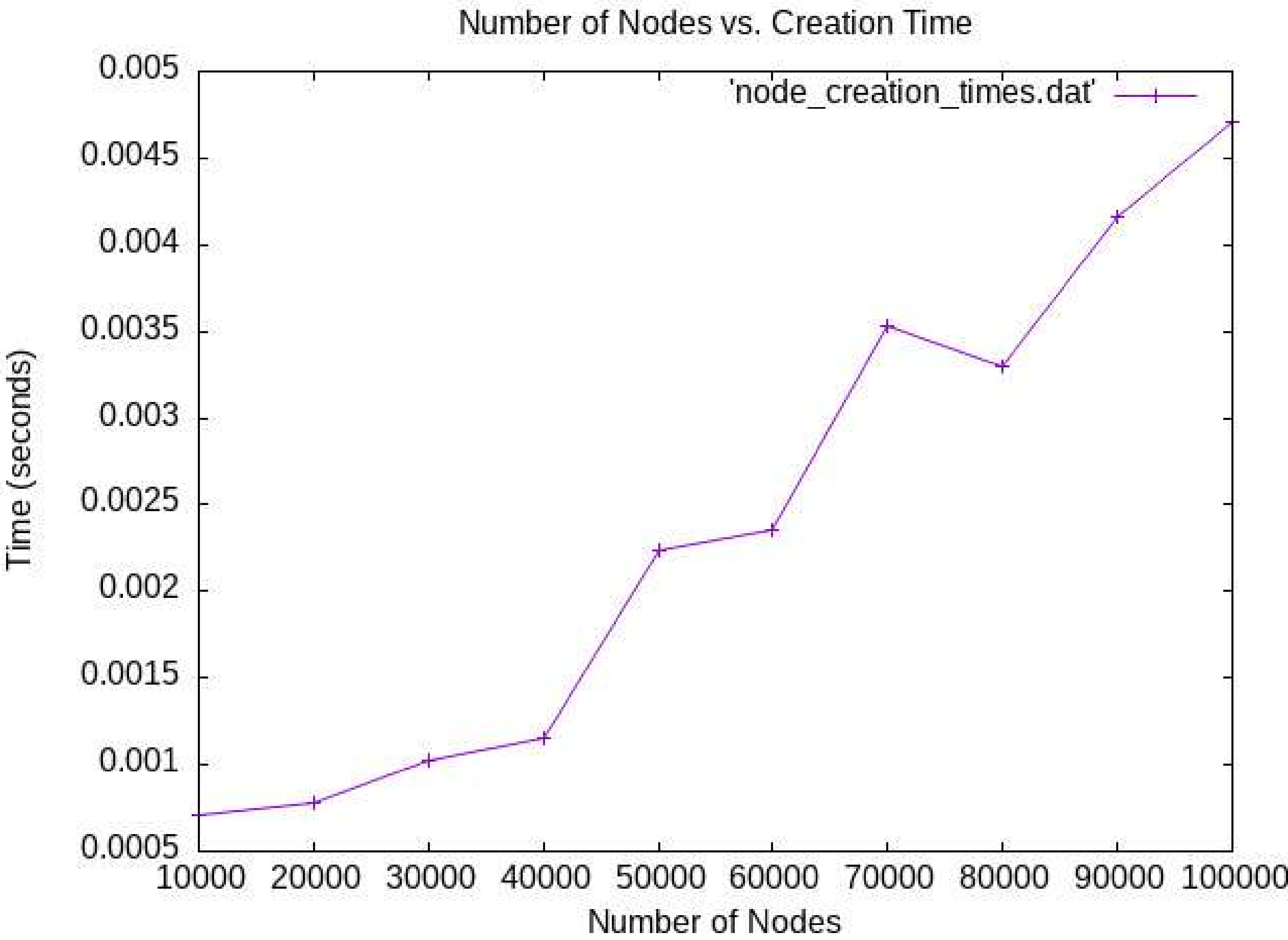
```
void measureMaxNodes(){
        const int increment ←10000;
        int count ← 0;
    try{
      While (true)(
          StudentRecord * head ← nullptr;
          studentRecord * tail ← nullptr;
        For (int i←0; i<increment; ++i){
          studentRecord ^ node ←new StudentRecord;
        IF (head = nullptr){
          head← tail ←node;
        } else{
          tail -> next ←node;
            tail ←node;
          }
       }
     count+= increment;
     O/P "SuccesFully Allocated" count
     StudentRecord* current ←head;
       while (current!=nullptr){
            studentRecord *next ← next(current);
            delete current;
            current←next;
         }
      }
  } Catch (bad_Alloc& e){
     O/P "Failed to allocate more Node. Maximum" count
     }
```

Obtimization, and system performonce measurement. Implementing such a system requires careful planning, attention to detail, and consideration for both functionality and efficiency.

The lesson learned can be applied to other areas of software developement that involve dynamic data management and performonce - critical operation

Lesson Learned

Doubly linked List offer flexibility for managing dynamic datasets with efficient operations. Modular design enhance code readability, maintainibility and scalability. measuring performone help in optimizing and understanding system constraints. Efficient file handing is crucial for maintaining responsivemess with large data. Comprehensive testing ensures robustness and reliability in diverse scenorios


Number of Nodes vs. Creation Time
'node_creation_times.dat'
Time (seconds)
0.005
0.0045
0.004
0.0035
0.003
0.0025
0.002
0.0015
0.001
0.0005
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of Nodes