Data Structures

Introduction

- Instructor:
  Asst. Prof. Sanem Kabadayı
- Teaching Assistant:
  Figen Şentürk
- Class Meeting Time:
  Tuesday 9:30-12:30 AM
- Course Web Site:
  http://ninova.itu.edu.tr/Ders/1657/Sinif/5910
  All announcements will be made only on the course web site. Students are expected to check this page regularly.

Course Evaluation Criteria

- For Data Structures, all sections will be graded on the same curve (including all exams and homework assignments).

Assessment	Percentage
Projects	30
Midterm	30
Final Exam	40

- The passing grade may vary from semester to semester depending on the class average. Generally, the passing grade is 50.

Requirements for Passing and Attendance Requirement

- Regulations Article 20- a) A minimum of 70% attendance at the classes and minimum of 80% attendance at independent applied classes such as laboratory and workshops is mandatory. Students who cannot fulfill the attendance requirement shall not be allowed to take the final examination at the end of the semester.
- Attendance may be taken at any point in the lecture. No additions can be made to the attendance list after that point.
- Students must turn in at least 2 out of the 3 assignments (an assignment counts as "turned in" only if it gets a grade above 30).
- Students should get at least 30 on the midterm.
- Average of homework assignments must be at least 55.
- Weighted average should be at least 40.
- Any student who gets a grade lower than the required grade on any of these assessments will fail the course with a grade of VF and not be allowed to take the final exam.

Important Note:

- I teach Computer Engineering!
- And this is Data Structures!

- Not Begging or Bargaining!

Prerequisites

- Students must have passed BIL105E Introduction to Scientific and Engineering Computing (C).
- **Knowledge of the C language** and having written programs (even if they were small examples) is an absolute must.
- Students who do not have knowledge of C have zero chance of being successful in Data Structures C.

Data Structures

- In computer science, a data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.
- A data structure is a mathematical and logical organization of data.
- For an algorithm to work efficiently, it has to use a well-designed data structure.
- A well-designed data structure allows a variety of critical operations to be performed using as little resources (e.g., execution time and memory space) as possible.

Data Structures

- In the design of large systems, it has been observed that the quality and performance of the final result depends heavily on choosing the best data structure.
- After the data structures are chosen, the designing of the algorithm becomes a relatively easier task.

Data Structures

- Array
- List
- Stack
- Queue
- Tree
- …

Introduction of the Lecture Example

PHONE BOOK

We will realize a phone book application that displays the numbers of recorded people, adds records, deletes records, updates records, and searches for records.

Lecture codes can be found on Ninova.

Lecture Examples

- In the sample code, the C programming language and structured programming will be used.
- For compatibility with new standards, we will make use of some novelties that the C++ language brings.
- These properties will be indicated where they are used.

Example:

C	C++

`printf("%d\n", number);`
`scanf("%d", &number);`
	`cout << number << endl;`
	`cin >> number;`

Phone Book

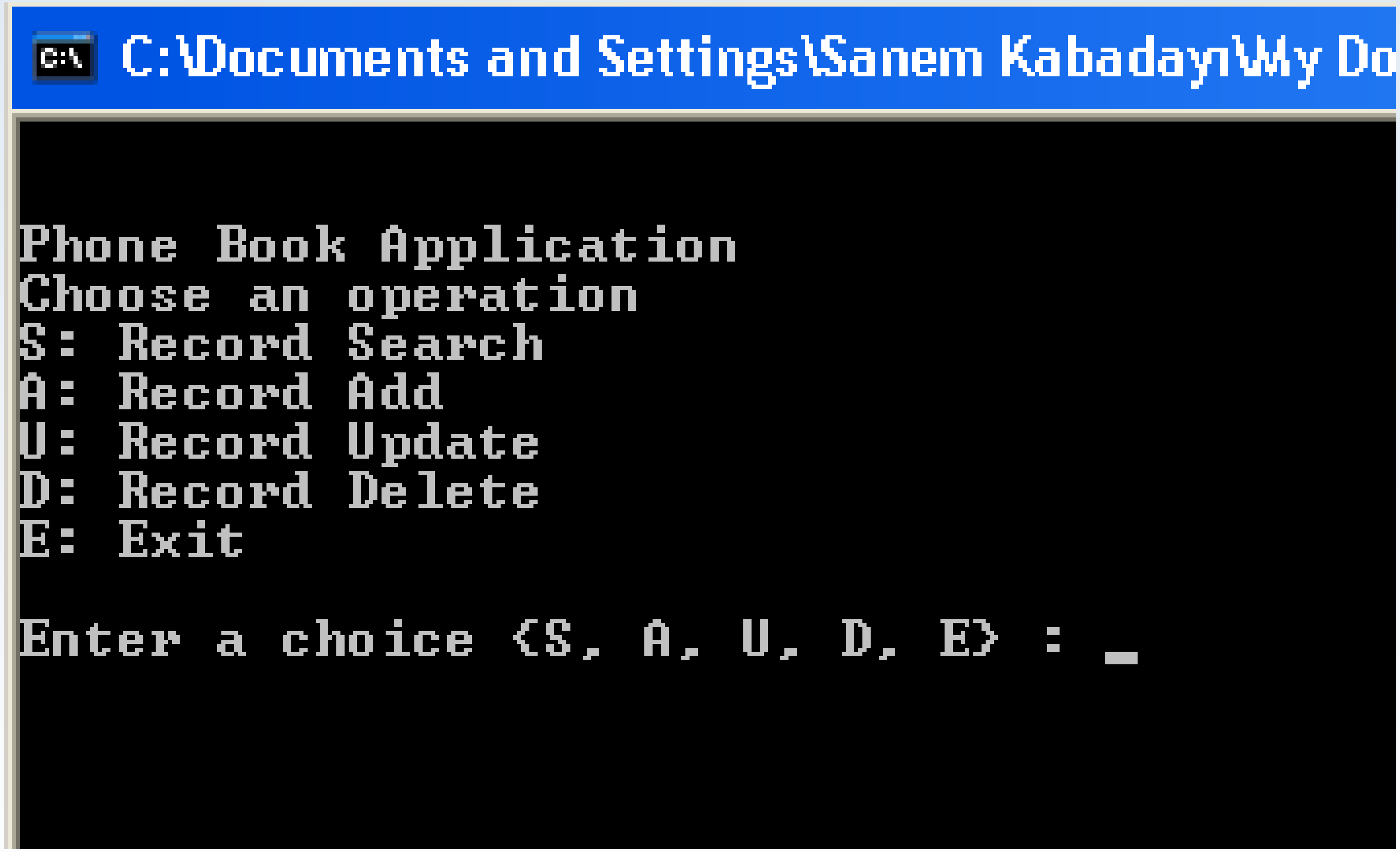
```
int main(){
  bool end = false;
  char choice;
  while (!end) {
      print_menu();
      cin >> choice;
      end = perform_operation(choice);
  }
  return EXIT_SUCCESS;
}
```

C++ bool data type

menu_print

```
void menu_print(){
  system("clear");
  cout << endl << endl;
  cout << "Phone Book Application" << endl;
  cout << "Choose an operation" << endl;
  cout << "S: Record Search" << endl;
  cout << "A: Record Add" << endl;
  cout << "U: Record Update" << endl;
  cout << "D: Record Delete" << endl;
  cout << "E: Exit" << endl;
  cout << endl;
  cout << "Enter a choice {S,A,U,D,E}: ";
}
```

menu_print



C:\Documents and Settings\Sanem Kabadayı\My Do

```
Phone Book Application
Choose an operation
S: Record Search
A: Record Add
U: Record Update
D: Record Delete
E: Exit

Enter a choice (S, A, U, D, E) : _
```

perform_operation

```
bool perform_operation(char choice){
    bool terminate=false;
    switch (choice) {
              case ‘S': case ‘s':
                            search_record();
                            break;
              case ‘A': case ‘a':
                            add_record();
                            break;
              case ‘U': case ‘u':
                            update_record();
                            break;
              case ‘D': case ‘d':
                            delete_record();
                            break;
              case ‘E': case ‘e':
                            cout << “Are you sure you want to exit the program? (Y/N):";
                            cin >> choice;
                            if(choice==‘Y' || choice==‘y')
                                          terminate=true;
                                          break;
              default:
                            cout << “Error: You have entered an invalid choice" << endl;
                            cout << “Please try again {S, A, U, D, E} :" ;
                            cin >> choice;
                            terminate = perform_operation(choice);
                            break;
    }
    return terminate;
}
```

perform_operation

```
bool perform_operation(char choice){
    bool terminate=false;
switch (choice) {
            case 'S': case 's':
                  search_record();
                  break;
            case 'A': case 'a':
                  add_record ();
                  break;
            case 'U': case 'u':
                  update_record();
                  break;
            case 'D': case 'd':
                  delete_record();
                  break;
```

perform_operation

```
bool perform_operation(char choice){
     bool terminate=false;
```

```
                    case ‘E': case ‘e':
                              cout << “Are you sure you want to exit the
                                        program? (Y/N):";
                              cin >> choice;
                              if(choice==‘Y' || choice==‘y')
                                        terminate=true;
                              break;
                    default:
                              cout << “Error: You have entered an invalid choice"
                                        << endl;
                              cout << “Please try again{S, A, U, D, E} :" ;
                              cin >> choice;
                              terminate = perform_operation(choice);
                              break;
          }
```

```
}
```

File Structure

- Data stored in structures in main memory are temporary and they disappear with the ending of the program.
- Files provide an environment where we can persistenly store data.
- Computers store files in secondary storage units (hard disks, magnetic disks, optical disks, ...).
- Performing operations in secondary storage units is slower than in main memory.

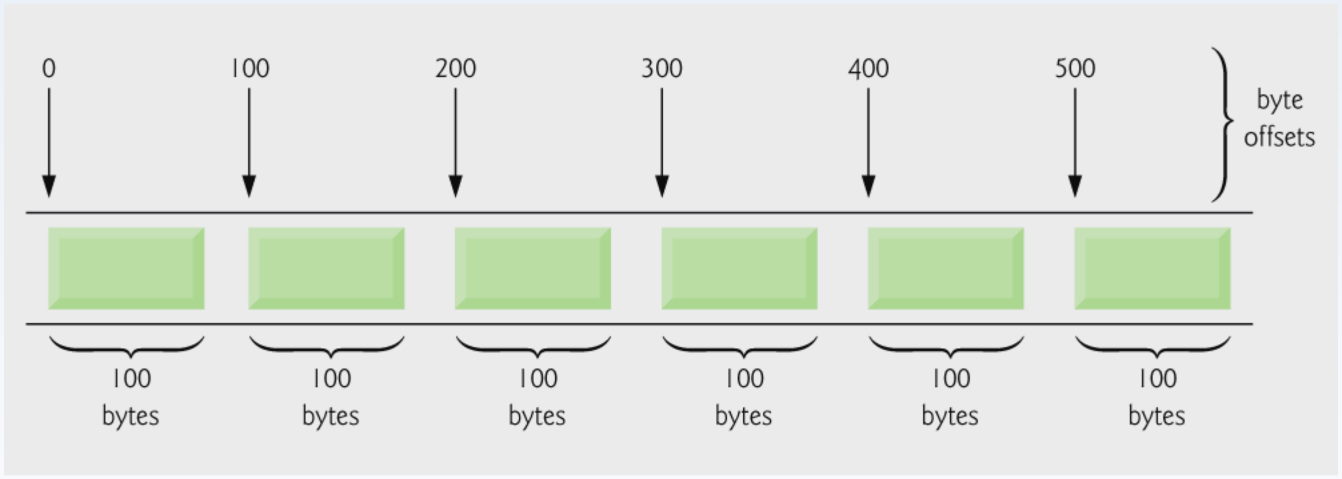
Record Structure

```
#define NAME_LENGTH 30
#define PHONENUM_LENGTH 15

struct Phone_Record{
  char name[NAME_LENGTH];
  char phonenum[PHONENUM_LENGTH];
};
```

Data Structure– 1. week File

- Random access files (reminder)
  - Provide direct access to records
  - Work on fixed-length records



C++ and Struct

- The struct structure of the C++ language, provides a natural capsule in defining abstract data types.
- Thus, the data type and the functions that define the operations that can be performed on this type are located in the same capsule and are logically linked.

C

```
typedef struct DataType{
   int array[10];
   int elementnumber;
} NewArrayType;

int ElementCount(NewArrayType
*d){
   return d-> elementnumber;
}
```

C++

```
struct DataType{
   int array[10];
   int elementnumber;
   int ElementCount();
};

int DataType::ElementCount(){
   return elementnumber;
}
```

Data Structure– 1. week File

```
“fileoperations.h”

struct File{
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
```

Data Structure

```
typedef File Datastructure;
Datastructure book;

int main(){
  book.create();
  bool end = false;
  char choice;
  while (!end) {
      print_menu();
      cin >> choice;
      end = perform_operation(choice);
      }
  book.close();
  return EXIT_SUCCESS;
}
```

Data Structure– 1. week File

“fileoperations.h”

```
struct File {
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
```

Data Structure “create”

```
void File::create(){
  filename=“phonebook.txt";
  phonebook = fopen( filename, "r+" );
  if(!phonebook){
      if(!(phonebook=fopen(filename,"w+"))){
          cerr << “Cannot open file" << endl;
          exit(1);
      }
  }
}
```

Data Structure– 1. week File

“fileoperations.h”

```
struct File {
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
```

Data Structure “close”

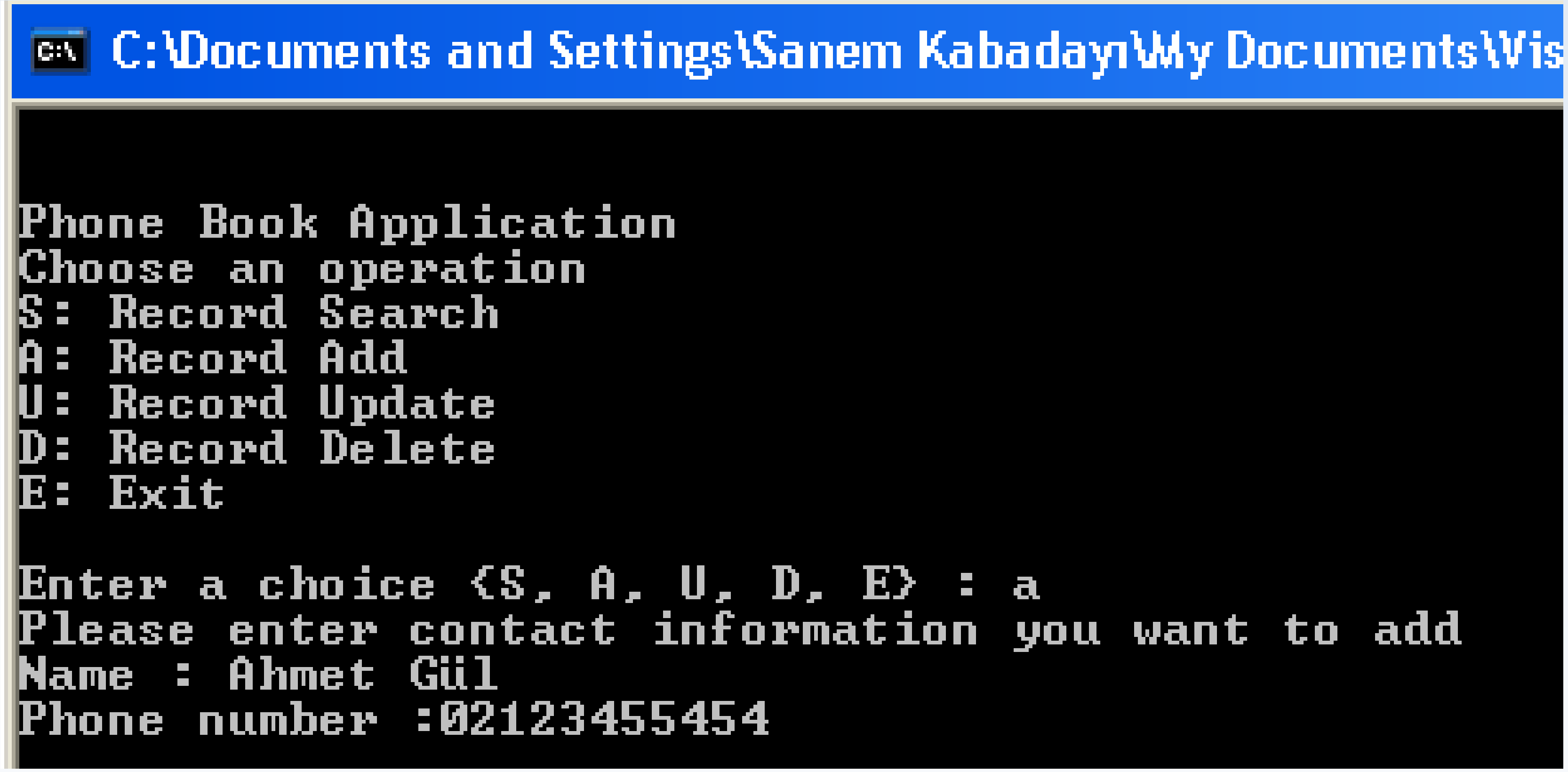
```
void File::close (){
  fclose(phonebook);
}
```

perform_operation

```
bool perform_operation(char choice){
   bool terminate=false;
   switch (choice) {
           case ‘S': case ‘s':
                 search_record();
                 break;
           case ‘A': case ‘a':
                 add_record();
                 break;
           case ‘U': case ‘u':
                 update_record();
                 break;
           case ‘D': case ‘d':
                 delete_record();
                 break;
           case ‘E': case ‘e':
                 cout << "Are you sure you want to exit the program? (Y/N):";
                 cin >> choice;
                 if(choice==‘Y' || choice==‘y')
                         terminate=true;
                         break;
           default:
                 cout << "Error: You have entered an invalid choice" << endl;
                 cout << "Please try again {S, A, U, D, E}  :" ;
                 cin >> choice;
                 terminate = perform_operation(choice);
                 break;
   }
   return terminate;
}
```

add_record

```
void add_record(){
  Phone_Record newrecord;
  cout << “Please enter contact information
      you want to add" << endl;
  cout << “Name : " ;
  cin.ignore(1000, '\n');
  cin.getline(newrecord.name,NAME_LENGTH);
  cout << “Phone number :";
  cin >> setw(PHONENUM_LENGTH)
      >>newrecord.phonenum;
  book.add(&newrecord);
  cout << “Record added" << endl;
  getchar();
};
```



```
C:\Documents and Settings\Sanem Kabadayı\My Documents\Vis
```

```
Phone Book Application
Choose an operation
S: Record Search
A: Record Add
U: Record Update
D: Record Delete
E: Exit

Enter a choice (S, A, U, D, E) : a
Please enter contact information you want to add
Name : Ahmet Gül
Phone number :02123455454
```

Data Structure – 1. week File

```
“fileoperations.h”

struct File{
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
```

add

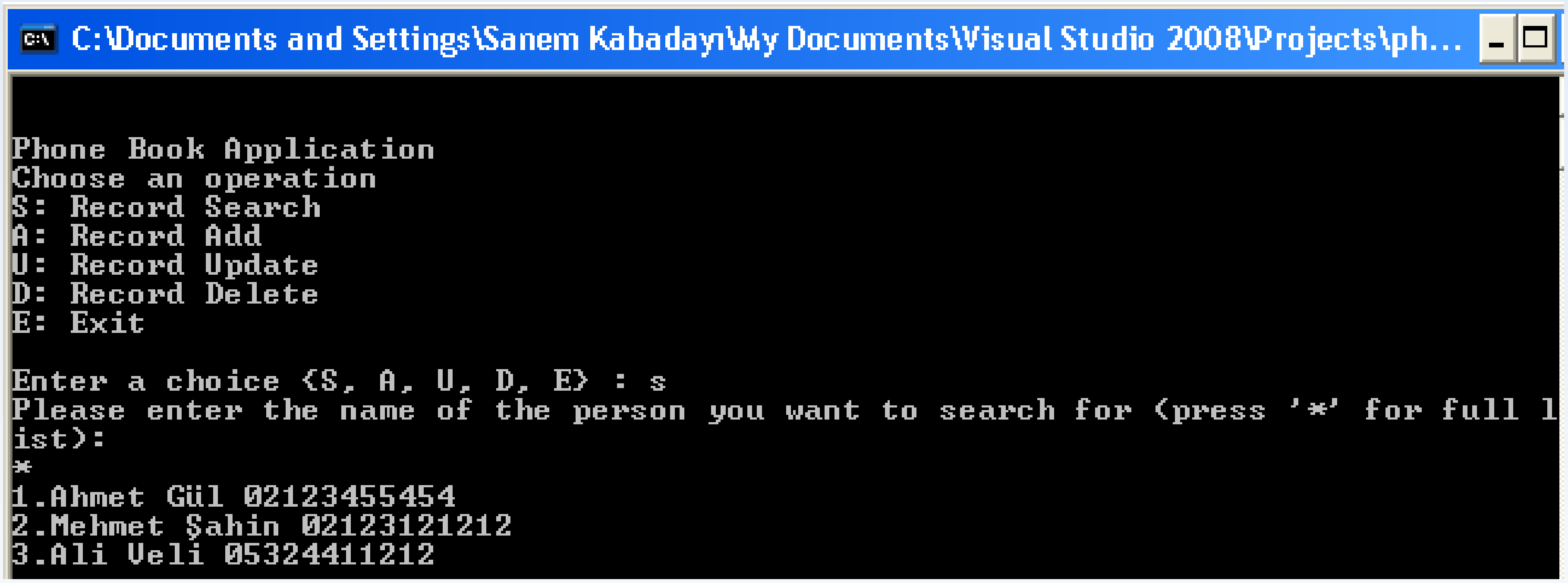
```
void File::add(Phone_Record *nrptr){
  fseek(phonebook, 0, SEEK_END);
  fwrite(nrptr, sizeof(Phone_Record), 1,
  phonebook);
}
```

perform_operation

```
bool perform_operation(char choice){
    bool terminate=false;
    switch (choice) {
            case ‘S': case ‘s':
                        search_record();
                        break;
            case ‘A': case ‘a':
                        add_record();
                        break;
            case ‘U': case ‘u':
                        update_record();
                        break;
            case ‘D': case ‘d':
                        delete_record();
                        break;
            case ‘E': case ‘e':
                        cout << "Are you sure you want to exit the program? (Y/N):";
                        cin >> choice;
                        if(choice==‘Y' || choice==‘y')
                                    terminate=true;
                                    break;
            default:
                        cout << "Error: You have entered an invalid choice" << endl;
                        cout << "Please try again {S, A, U, D, E}  :" ;
                        cin >> choice;
                        terminate = perform_operation(choice);
                        break;
    }
    return terminate;
}
```

search_record

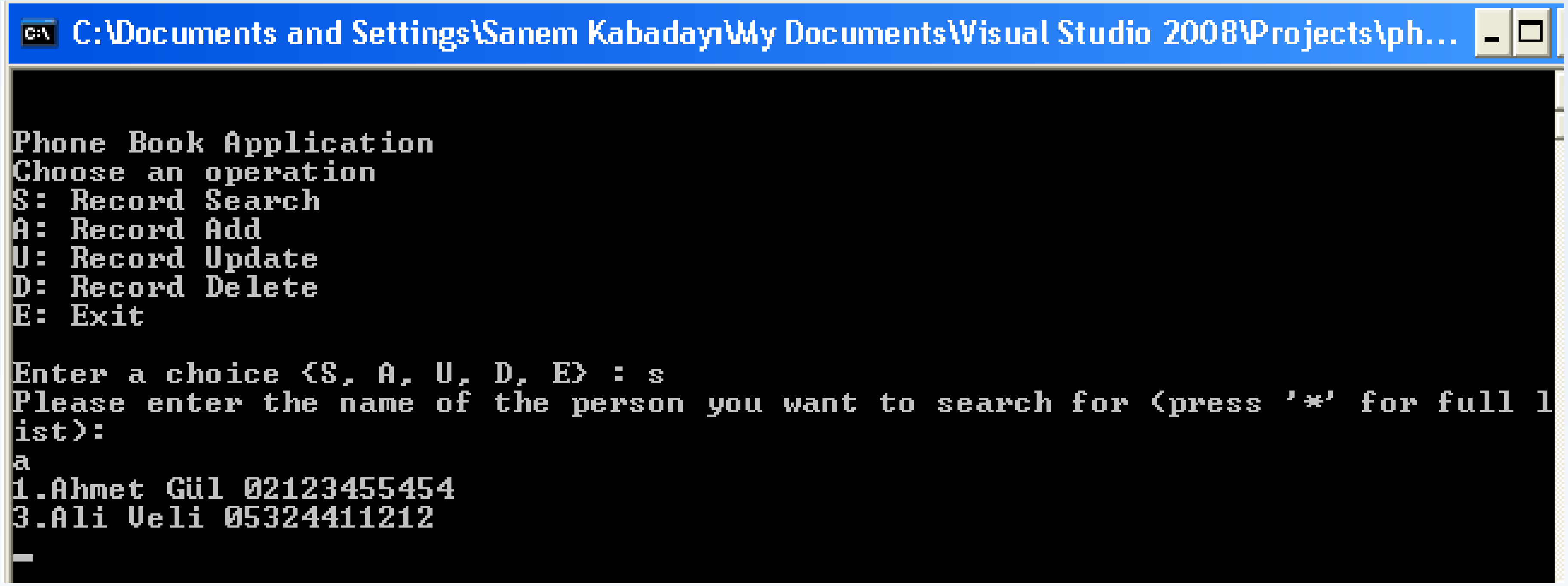
```
void search_record(){
  char name[NAME_LENGTH];
  cout << “Please enter the name of the person
  you want to search for (press'*'for full
  list):" << endl;
  cin.ignore(1000, '\n');
  cin.getline(name,NAME_LENGTH);
  if(book.search(name)==0){
      cout << “Could not find a record
      matching your search criteria" << endl;
  }
  getchar();
};
```



C:\Documents and Settings\Sanem Kabadayı\My Documents\Visual Studio 2008\Projects\ph...

```
Phone Book Application
Choose an operation
S: Record Search
A: Record Add
U: Record Update
D: Record Delete
E: Exit

Enter a choice {S, A, U, D, E} : s
Please enter the name of the person you want to search for (press '*' for full l
ist):
*
1.Ahmet Gül 02123455454
2.Mehmet Şahin 02123121212
3.Ali Veli 05324411212
```



C:\Documents and Settings\Sanem Kabadayı\My Documents\Visual Studio 2008\Projects\ph...

```
Phone Book Application
Choose an operation
S: Record Search
A: Record Add
U: Record Update
D: Record Delete
E: Exit

Enter a choice {S, A, U, D, E} : s
Please enter the name of the person you want to search for (press '*' for full l
ist):
a
1.Ahmet Gül 02123455454
3.Ali Veli 05324411212

_
```

Data Structure – 1. week File

```
“fileoperations.h”

struct File{
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
```

```
int File::search(char *desired){
   Phone_Record k;
   int index=0;
   bool all=false;
   int found=0;
   if(strcmp(desired,"*")==0)
        all=true;
   fseek(phonebook, 0, SEEK_SET);
   while(!feof(phonebook)){
        index++;
        fread( &k, sizeof (Phone_Record), 1, phonebook);

        if(feof(phonebook)) break;
        if(all || strnicmp(k.name,desired,strlen(desired))==0){
               cout << index <<"."”<< k.name <<" “<< k.phonenum
                    << endl;
               found++;
        }
   }
   return found;
```

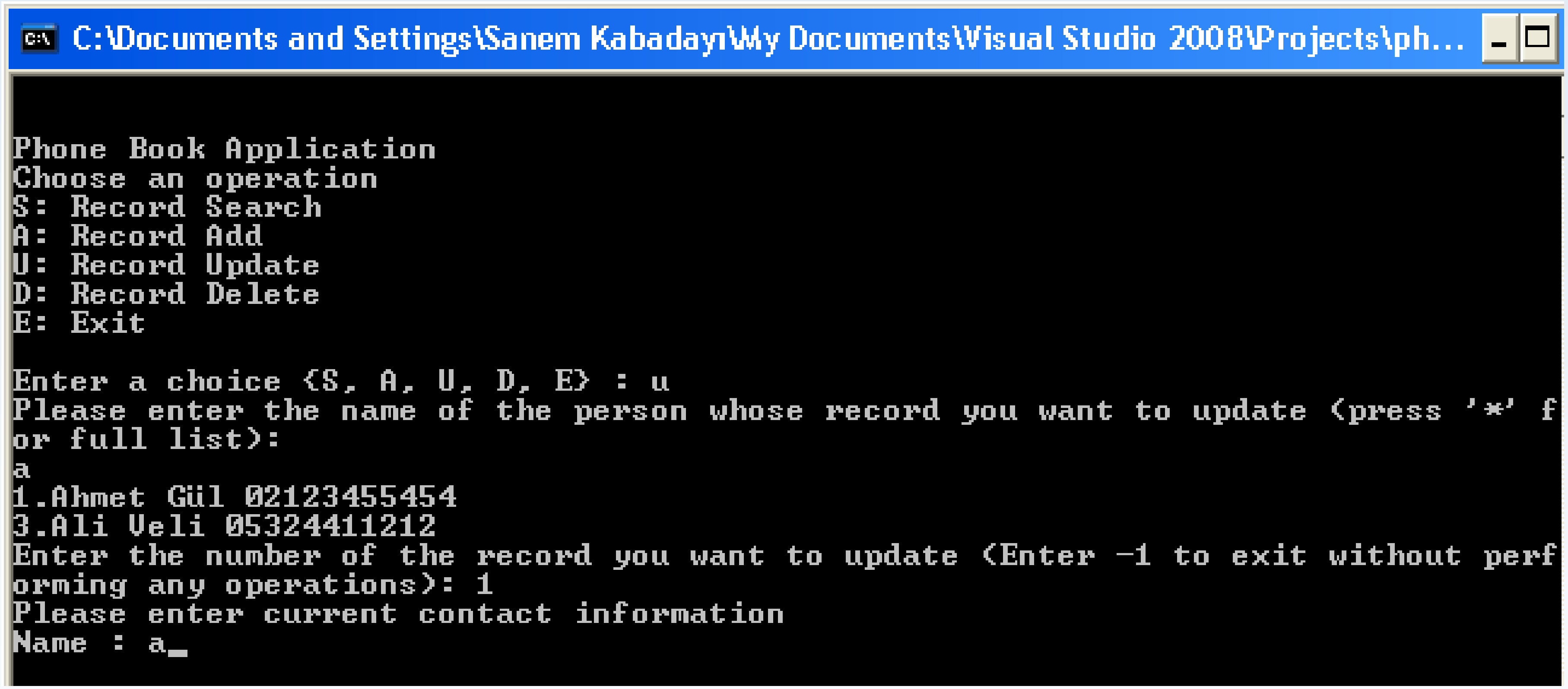
perform_operation

```
bool perform_operation(char choice){
    bool terminate=false;
    switch (choice) {
            case 'S': case 's':
                    search_record();
                    break;
            case 'A': case 'a':
                    add_record();
                    break;
            case 'U': case 'u':
                    update_record();
                    break;
            case 'D': case 'd':
                    delete_record();
                    break;
            case 'E': case 'e':
                    cout << "Are you sure you want to exit the program? (Y/N):";
                    cin >> choice;
                    if(choice=='Y' || choice=='y')
                            terminate=true;
                            break;
            default:
                    cout << "Error: You have entered an invalid choice" << endl;
                    cout << "Please try again {S, A, U, D, E}  :" ;
                    cin >> choice;
                    terminate = perform_operation(choice);
                    break;
    }
    return terminate;
}
```

update_record

```
void update_record(){
  char name[NAME_LENGTH];
  int choice;
  cout << “Please enter the name of the person whose record you want
              to update (press'*'for full list):" << endl;
  cin.ignore(1000, '\n');
  cin.getline(name,NAME_LENGTH);
  int personcount=book.search(name);
  if(personcount==0){
       cout << "Could not find a record matching your search criteria"
  << endl;
  }
  else {
        if (personcount==1){
              cout << "Record found." << endl;
              cout << " If you want to update this record please enter
                         its number (Enter -1 to exit without
                         performing any operations): " ;
       }else cout << “Enter the number of the record you want to
       update (Enter -1 to exit without performing any operations): ";
```

```
        cin >> choice;
        if(choice==-1) return;
        Phone_Record newrecord;
        cout << “Please enter current contact
                    information" << endl;
        cout << “Name : ";
        cin.ignore(1000, '\n');
        cin.getline(newrecord.name,NAME_LENGTH);
        cout << “Phone number :";
        cin >> setw(PHONENUM_LENGTH) >> newrecord.phonenum;
        book.update(choice,&newrecord);
        cout << “Record successfully updated" << endl;
    }
    getchar();
};
```



```
C:\Documents and Settings\Sanem Kabadayı\My Documents\Visual Studio 2008\Projects\ph...
```

```
Phone Book Application
Choose an operation
S: Record Search
A: Record Add
U: Record Update
D: Record Delete
E: Exit

Enter a choice (S, A, U, D, E) : u
Please enter the name of the person whose record you want to update (press '*' f
or full list):
a
1.Ahmet Gül 02123455454
3.Ali Veli 05324411212
Enter the number of the record you want to update (Enter -1 to exit without perf
orming any operations): 1
Please enter current contact information
Name : a_
```

```
void File::update(int recordnum,Phone_Record
  *nrptr){
  if(fseek(phonebook,
       sizeof(Phone_Record)*(recordnum-1),
      SEEK_SET) == 0)
     fwrite(nrptr, sizeof(Phone_Record), 1,
          phonebook);
}
```

perform_operation

```
bool perform_operation(char choice){
   bool terminate=false;
   switch (choice) {
           case ‘S': case ‘s':
                   search_record();
                   break;
           case ‘A': case ‘a':
                   add_record();
                   break;
           case ‘U': case ‘u':
                   update_record();
                   break;
           case ‘D': case ‘d':
                   delete_record();
                   break;
           case ‘E': case ‘e':
                   cout << "Are you sure you want to exit the program? (Y/N):";
                   cin >> choice;
                   if(choice==‘Y' || choice==‘y')
                           terminate=true;
                           break;
           default:
                   cout << "Error: You have entered an invalid choice" << endl;
                   cout << "Please try again {S, A, U, D, E}  :" ;
                   cin >> choice;
                   terminate = perform_operation(choice);
                   break;
   }
   return terminate;
}
```

delete_record

```
void delete_record(){
  char name[NAME_LENGTH];
  int choice;
  cout << “Please enter the name of the person whose
          record you want to delete (press'*'for full
          list):" << endl;
  cin.ignore(1000, '\n');
  cin.getline(name,NAME_LENGTH);
  int personcount=book.search(name);
  if(personcount==0){
      cout << " Could not find a record matching your
            search criteria " << endl;
  }
  else {
     if (personcount==1){
       cout << "Record found." << endl;
       cout << “If you want to delete this record
                please enter its number (Enter -1 to exit
                without performing any operations): " ;
    }
    else cout << “Enter the number of the record you want
                  to delete (Enter -1 to exit without
                  performing any operations): " ;
    cin >> choice;
    if(choice==-1) return;
    book.remove(choice);
    cout << "Record deleted" <<endl;
  }
  getchar();
};
```

```
void File::remove(int recordnum){
Phone_Record emptyrecord={"",""};
if(fseek(phonebook,
  sizeof(Phone_Record)*(recordnum-
  1),SEEK_SET)==0)
  fwrite(&emptyrecord,sizeof(Phone_Record),
          1,phonebook);
}
```

phoneprog.cpp

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
#include <ctype.h>
#include "fileoperations.h"

using namespace std;

typedef File Datastructure;
Datastructure book;

void print_menu();
bool perform_operation(char);
void search_record();
void add_record();
void delete_record();
void update_record();

int main(){ ...
```

record.h

```
#define NAME_LENGTH 30
#define PHONENUM_LENGTH 15

struct Phone_Record{
  char name[NAME_LENGTH];
  char phonenum[PHONENUM_LENGTH];
};
```

fileoperations.h

```
#ifndef FILEOPERATIONS_H
#define FILEOPERATIONS_H
#include <stdio.h>
#include "record.h"

struct File{
  char *filename;
  FILE *phonebook;
  void create();
  void close();
  void add(Phone_Record *);
  int search(char []);
  void remove(int recordnum);
  void update(int recordnum, Phone_Record *);
};
#endif
```

fileoperations.cpp

```
#include "fileoperations.h"
#include <iostream>
#include <stdlib.h>
#include <string.h>

using namespace std;

void File::add(Phone_Record *nrptr){
  fseek(phonebook, 0, SEEK_END);
  fwrite(nrptr, sizeof(Phone_Record), 1,
  phonebook);
}

void File::create(){     ...
```

Recitation: To do

- Explanation of compilation steps and compilation operations

Practice Exercise

Realize:

1. The real deletion operation on the lecture example