OBJECT-ORIENTED PROGRAMMING SYSTEM ASSIGNMENT

Name -Atharv Dayanand Shet Govekar
Branch -Computer Engineering
Div - A
Batch – I
Roll-No.- 23B-CO-010
Date – 1 August 2024

GOA COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT: - OOPS

FACULTY: - Prof. AMIT P. PATIL **CLASS: - SE Comp (III)**

PLATFORM: - Dev C++/VS 2010 **YEAR: - 1-7-24 to 12/11/24**

Date of Announcement: 25-7-24 **Date of Submission: 8-8-24**

Assignment No 2

1.

An election is contested by five candidates. The candidates are numbered 1 to 5 and the voting is done by marking the candidate number on the ballot paper. Write a program to read the ballots and ***count*** *the votes cast for each candidate using an array variable count. In case, a number read is outside the range 1 to 5, the ballot should be considered as a 'spoilt ballot', and the program should also count the number of spoilt ballots.*

2.

A cricket team has the following table of batting figures for a series of test matches:

Player's name	Runs	Innings	Times not out
Sachin	8430	230	18
Saurav	4200	130	9
Rahul	3350	105	11
.	.	.	.
.	.	.	.

Write a program to read the figures set out in the above form, to calculate the batting averages and to print out the complete table including the averages.

Note : use setw() manipulator to format the output.
Batting avg = runs/ times batsman is out

3.

Write programs to evaluate the following functions to 0.0001% accuracy.

(a) $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots\cdots$

(b) $SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + \cdots\cdots$

4. With diagram explain how the memory allocation is done for objects. Also state the mechanism to perform dynamic memory allocation to objects.

5. Write a program to demonstrate the dynamic memory allocation for a matrix. Read the matrix elements and display the same.

1)

INPUT:

```
#include <iostream>
using namespace std ;

int votes (int voter[],int i ){
   int choice ;

cout<<"The candidates of election are \n";
```

```
cout <<"1.Candidate 1\n" ;
cout <<"2.Candidate 2\n" ;
cout <<"3.Candidate 3\n" ;
cout <<"4.Candidate 4\n" ;
cout <<"5.Candidate 5\n" ;
cout <<"Enter your vote\n";
for (int j=0;j<i;j++){

cin>>choice ;
voter[j] = choice ;
}
return choice ;
}
void election (int voter[],int i) {
     int count[5]={0,0,0,0,0} ;
   int spoilt_vote=0 ;
   for (int j=0;j<i;j++){
 int vote = voter[j] ;
switch (vote){
case 1 : count[0] ++ ;
         break ;
case 2 : count[1] ++ ;
         break ;
case 3 : count[2] ++ ;
         break ;
case 4 : count[3] ++ ;
         break ;
case 5 : count[4] ++ ;
         break ;

default : spoilt_vote ++ ;
        break ;
} }
for(int i=0;i<5;i++) {
   cout <<"The votes for Candidate "<<i+1<<" are "<<count[i]<<" votes"<<endl ;
}
   cout <<"The number of spoilt votes are "<<spoilt_vote<<" votes"<<endl ;
return ;
}
int main (){
   int i ;
   cout<<"Enter the number of voters in the city \n" ;
   cin>>i ;
int voter[100];
votes(voter,i);
election(voter,i) ;
   return 0 ;
}
```

OUTPUT -

```
Enter the number of voters in the city
21
The candidates of election are
1.Candidate 1
2.Candidate 2
3.Candidate 3
4.Candidate 4
5.Candidate 5
Enter your vote
4
3
2
1
6
7
4
3
2
1
1
1
3
4
5
6
4
3
2
2
1
The votes for Candidate 1 are 5 votes
The votes for Candidate 2 are 4 votes
The votes for Candidate 3 are 4 votes
The votes for Candidate 4 are 4 votes
The votes for Candidate 5 are 1 votes
The number of spoilt votes are 3 votes
```

2)

INPUT-

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

class player {
private:
   string name;
```

```
    int runs;
    int innings;
    int notouting;
    float average;

public:
    void input() {
        cin.ignore();
        cout << "Enter the name of the player\n";
        getline(cin, name);
        cout << "Enter the runs he scored in his career\n";
        cin >> runs;
    label:
        cout << "Enter the innings he has played \n";
        cin >> innings;
        cout << "Enter the number of times he remained not out \n";
        cin >> notouting;
        if (notouting > innings) {
            cout << "Enter valid data\n";
            goto label;
        }
        average = static_cast<float>(runs) / (innings - notouting);
        return;
    }

    void display() {
        cout << setw(25) << left << name
             << setw(10) << right << runs
             << setw(10) << right << innings
             << setw(17) << right << notouting
             << setw(10) << right << fixed << setprecision(2) << average
             << endl;
    }
};

int main() {
    int n;
    player s[100];
    cout << "Enter the number of players\n";
    cin >> n;

    for (int i = 0; i < n; i++) {
        s[i].input();
    }

    cout << "\n" << setw(25) << left << "Player's name"
         << setw(10) << right << "Runs"
         << setw(10) << right << "Innings"
```

```
        << setw(17) << right << "Times not out"
        << setw(10) << right << "Average"
        << endl;

    for (int i = 0; i < n; i++) {
        s[i].display();
    }
    return 0;
}
```

OUTPUT –

```
Enter the number of players
4
Enter the name of the player
Aditya Malik
Enter the runs he scored in his career
6543
Enter the innings he has played
453
Enter the number of times he remained not out
231
Enter the name of the player
Ravi Pushkar
Enter the runs he scored in his career
5644
Enter the innings he has played
343
Enter the number of times he remained not out
23
Enter the name of the player
Rajesh Ojha
Enter the runs he scored in his career
6738
Enter the innings he has played
566
Enter the number of times he remained not out
42
Enter the name of the player
Samir Tendulkar
Enter the runs he scored in his career
9087
Enter the innings he has played
564
Enter the number of times he remained not out
336

Player's name                Runs    Innings    Times not out    Average
Aditya Malik                 6543        453              231      29.47
Ravi Pushkar                 5644        343               23      17.64
Rajesh Ojha                  6738        566               42      12.86
Samir Tendulkar              9087        564              336      39.86
```

3)

INPUT-

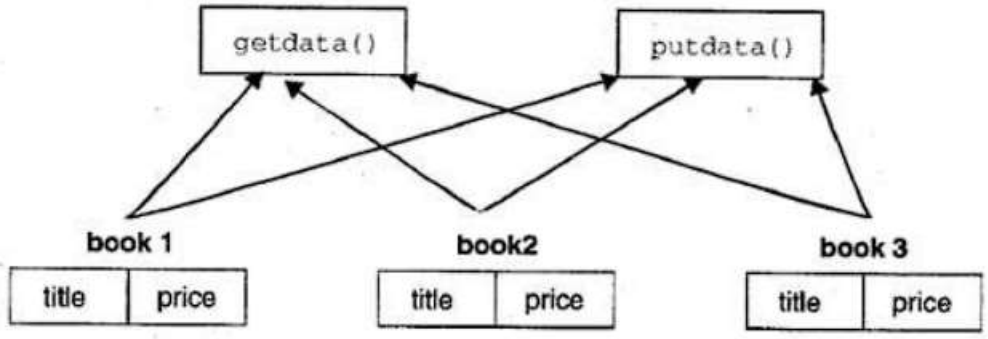
```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

void sinfunc() {
   int n;
   float sinx = 0, x;

   cout << "Enter the number of terms you want in the series\n";
   cin >> n;
   cout << "Enter the value of x\n";
   cin >> x;

   for (int i = 0; i < n; i++) {
      float term = pow(x, 2 * i + 1);
      float fact = 1;
      for (int j = 1; j <= 2 * i + 1; j++) {
         fact *= j;
      }
      if (i % 2 == 0) {
         sinx += term / fact;
      } else {
         sinx -= term / fact;
      }
   }

   cout << "The value of sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ... till "
       << n << " terms is " << fixed << setprecision(4) << sinx << endl;
   return;
}

void sum() {
   int n;
   float sum = 0;

   cout << "Enter the number of terms you want in the series\n";
   cin >> n;

   for (int i = 1; i <= n; i++) {
      float term = pow(1.0 / i, i);
      sum += term;
```

```
    }

    cout << "The value of SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ... till "
        << n << " terms is " << fixed << setprecision(4) << sum << endl;
    return;
}

int main() {
    int choice;
    do {
        cout << "\n\nEnter the operation you want to perform\n";
        cout << "1. sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ...\n";
        cout << "2. SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ...\n";
        cout << "3. Quit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                sinfunc();
                break;
            case 2:
                sum();
                break;
            case 3:
                break;
            default:
                cout << "Enter a valid option\n";
                break;
        }
    } while (choice != 3);
    return 0;
}
```

OUTPUT –

```
Enter the operation you want to perform
1. sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ...
2. SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ...
3. Quit
Enter choice: 1
Enter the number of terms you want in the series
4
Enter the value of x
2
The value of sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ... till 4 terms is 0.9079


Enter the operation you want to perform
1. sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ...
2. SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ...
3. Quit
Enter choice: 2
Enter the number of terms you want in the series
6
The value of SUM = 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ... till 6 terms is 1.2913
```

4)



Memory Allocation for the Objects of the Class book

Dynamic memory allocation is a technique that allows programs to request and manage memory while they are running. Unlike fixed memory allocation, which reserves a specific amount of memory at the start, dynamic allocation lets programs adjust their memory usage based on current needs.

Dynamic memory allocation is used when the size or number of data structures can't be determined before the program starts. This method helps manage memory more efficiently by allocating memory only when needed and releasing it when it's no longer in use.

Mechanisms of Dynamic Memory Allocation in C++

1. **Requesting and Allocating Memory:**
    - **Using malloc and free:** In C, memory is requested using malloc, which allocates a specified amount of memory and returns a pointer to it. For example, malloc(sizeof(MyClass)) allocates memory for one MyClass object. Memory is then released using free, which deallocates the memory block previously allocated by malloc.
    - Using new and delete: In C++, new is preferred for allocating memory. It not only allocates memory but also calls the constructor of the class, initializing the object. For instance, new MyClass() allocates memory and initializes a MyClass object. Memory is released with delete, which also calls the destructor of the class to properly clean up resources. For arrays, new[] and delete[] are used, which handle multiple objects and their initialization and cleanup.
    - new and delete handle object construction and destruction, ensuring that resources are properly managed and initialized. This is crucial for C++ where constructors and destructors are used to manage resource allocation and cleanup.

- new and delete provide better type safety compared to malloc and free. They ensure that the correct type is used and that the appropriate constructor and destructor are called, reducing the risk of errors.

  new can throw exceptions if memory allocation fails, while malloc returns a null pointer. This can lead to more robust error handling in C++ programs.

2. **Managing Memory:** The system maintains information about allocated and free memory blocks. When memory is released using delete, it is returned to the heap and made available for future allocations. Proper management helps prevent issues like memory leaks and ensures efficient use of memory.

Advantages

Dynamic memory allocation provides:

- **Flexibility:** Memory is allocated based on actual needs, allowing programs to adapt to varying requirements.
- **Efficiency:** Memory is used more effectively by allocating only what is needed and releasing it when no longer in use.

5)
INPUT-

```
#include <iostream>
using namespace std ;

void matrix (){
int c,r,t;
cout<<"Enter the number of rows and columns\n" ;
cin>>r ;
cin>>c ;
int **p = new int*[r] ;  //Use of double pointer to dynamically allocate rows using new
for(int i=0;i<r;i++){
   p[i]=new int[c];  //Use of new to dynamically allocate columns
}
cout <<"Enter the matrix elements\n";
for(int i=0;i<r;i++){
for(int j=0;j<c;j++){
cin>>t ;       //Inputing matrix elements
p[i][j]=t ;
}
}

cout <<"The matrix is: \n";
for(int i=0;i<r;i++){
   cout<<"|" ;
for(int j=0;j<c;j++){
cout<<" "<<p[i][j];    //Displaying matrix elements
}
cout<<" |" ;
cout<<endl ;
}
}

int main (){
matrix ();
   return 0 ;
}
```

OUTPUT-

```
Enter the number of rows and columns
3
3
Enter the matrix elements
6
5
8
2
1
7
3
4
0
The matrix is:
| 6 5 8 |
| 2 1 7 |
| 3 4 0 |
```