Object Oriented Programming (OOP)
Lecture No. 13

Review

- Static data members
- Static member functions
- Array of objects

Pointer to Objects

- Pointer to objects are similar as pointer to built-in types
- They can also be used to dynamically allocate objects

Example

```
class Student{
…
public:
   Studen();
   Student(char * aName);
   void setRollNo(int aNo);
};
```

Example

```
int main(){
   Student obj;
   Student *ptr;
   ptr = &obj;
   ptr->setRollNo(10);
   return 0;
}
```

Allocation with new Operator

- new operator can be used to create objects at runtime

Example

```
int main(){
   Student *ptr;
   ptr = new Student;
   ptr->setRollNo(10);
   return 0;
}
```

Example

```
int main(){
   Student *ptr;
   ptr = new Student("Ali");
   ptr->setRollNo(10);
   return 0;
}
```

Example

```
int main()
{
   Student *ptr = new Student[100];
   for(int i = 0; i < 100;i++)
   {
      ptr->setRollNo(10);
   }
   return 0;
}
```

Breakup of new Operation

- new operator is decomposed as follows
  - Allocating space in memory
  - Calling the appropriate constructor

Case Study

Design a class date through which user must be able to perform following operations

- Get and set current day, month and year
- Increment by x number of days, months and year
- Set default date

Attributes

- Attributes that can be seen in this problem statement are
  - Day
  - Month
  - Year
  - Default date

Attributes

- The default date is a feature shared by all objects
  - This attribute must be declared a static member

Attributes in Date.h

```
class Date
{
   int day;
   int month;
   int year;
   static Date defaultDate;
…
};
```

Interfaces

- getDay
- getMonth
- getYear
- setDay
- setMonth
- setYear
- addDay
- addMonth
- addYear
- setDefaultDate

Interfaces

- As the default date is a static member the interface setDefaultDate should also be declared static

Interfaces in Date.h

```
class Date{
…
public:
   void setDay(int aDay);
   int getDay() const;
   void addDay(int x);
   …
…
};
```

Interfaces in Date.h

```
class Date{
…
public:
   static void setDefaultDate(
int aDay,int aMonth, int aYear);
…
};
```

Constructors and Destructors in Date.h

```
   Date(int aDay = 0,
    int aMonth= 0, int aYear= 0);

   ~Date(); //Destructor
};
```

Implementation of Date Class

- The static member variables must be initialized

Date Date::defaultDate (07,3,2005);

Constructors

```
Date::Date(int aDay, int aMonth,
                    int aYear)    {
     if(aDay==0) {
          this->day = defaultDate.day;
     }
     else{
          setDay(aDay);
     }
     //similarly for other members
}
```

Destructor

- We are not required to do any house keeping chores in destructor

```
Date::~Date
{
}
```

Getter and Setter

```
void Date::setMonth(int a){
   if(a > 0 && a <= 12){
      month = a;
}
int getMonth() const{
   return month;
}
```

addYear

```
void Date::addYear(int x){
   year += x;
   if(day == 29 && month == 2
      &&   !leapyear(year)){
      day = 1;
      month = 3;
   }
}
```

Helper Function

```
class Date{
…
private:
   bool leapYear(int x) const;
…
};
```

Helper Function

```
bool Date::leapYear(int x) const{
   if((x%4 == 0 && x%100 != 0)
       ||  (x%400==0)){
      return true;
   }
   return false;
}
```

setDefaultDate

```
void Date::setDefaultDate(
   int d, int m, int y){
   if(d >= 0 && d <= 31){
      day = d;
   }
   …
}
```

Recap