Class Member Functions

This lesson teaches what member functions are and how they can be used to access private variables

We'll cover the following

Defining Member Functions

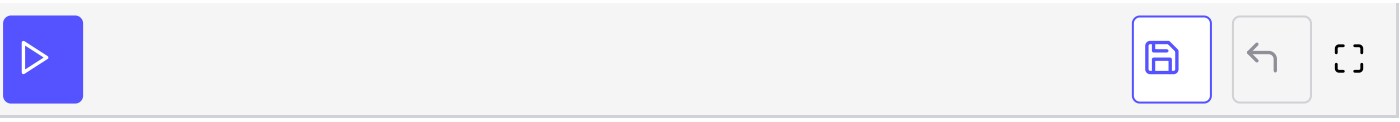
Member functions are *declared* in the class declaration.

```
class DayOfYear {
public:
    void output(); //member function
    int month;
    int day;
};
```

Member function *definitions* identify the class in which the function is a *member*.

```
class DayOfYear {
public:
    void output(); //member function
    int month;
    int day;
};

void DayOfYear::output(){   //indicates function output is member of DayOfYear class
  cout << "Month =  " << month <<endl;
  cout << "Day =  " << day <<endl;
}

int main(){
  DayOfYear birthday, today ; //declaring multiple objects of class DayOfYear
                              //today and birthday objects have their own verions
                              //of class member variables

  birthday.month = 8;    //accessing and setting month variable for object birthday
```

```
  birthday.day = 5;    //accessing and setting day variable for object birthday

  cout << "Your birthday date is: "<<endl;

  birthday.output();    //calling the member function output

  cout << "Today's date is: "<<endl;

  today.month = 12;  //accessing and setting month variable for object today

  today.day = 4;    //accessing and setting day variable for object today

  today.output();  //since output() is public it is directly accessible in main()

}
```



Setters and Getters

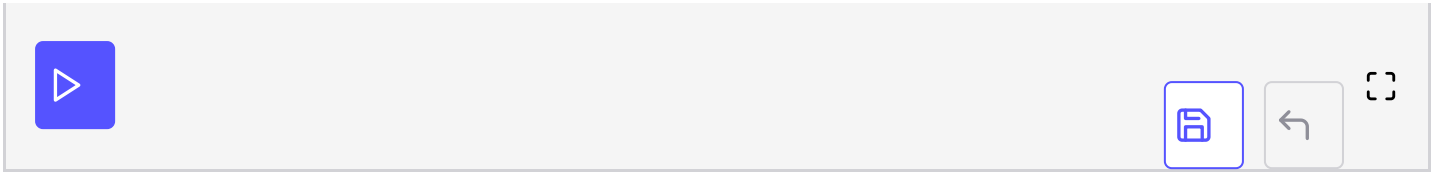
As we discussed, earlier `private` *member* variables cannot be accessed directly in any other function.

In order to *access* or *change* their values, we need to *define* `public` *member* functions. These *functions* can be used to *set* the values of the `private` variables as well as to *get* their values since being `private` these members cannot be accessed directly.

Take a look at the example below to understand this better.

```
//Class with Private Members
//Program to demonstrate the class DayOfYear.
#include <iostream>
using namespace std;

class DayOfYear
{
public:
          int myVar;
    void output( );

    void set(int new_month, int new_day);
    //Precondition: new_month and new_day form a possible date.
    //Postcondition: The date is reset according to the arguments.

    int get_month( );
    //Returns the month, 1 for January, 2 for February, etc.

    int get_day( );
    //Returns the day of the mont
private:  //private variables not accessible directly in main
    void check_date( );  //checks the date
    int month;
    int day;
```

```
    int day;
};

int main( )
{
    DayOfYear today, birthday; //making two objects of DayOfYear class
    //setting today object's month and day
    today.set(11,23);
    cout << "Today's date is ";
    today.output( );    //calling output to display today object's month and day

    //setting birthday object's month and day
    birthday.set(3, 21);  //try setting these values same as one passed for today.
                          //also try passing invalid values here and then run code
    cout << "Birthday date is ";
    birthday.output( );  //calling output to display birthday object's month and day

    if (today.get_month() == birthday.get_month() //if condition to check if today object's month/
       && today.get_day() == birthday.get_day() ) //equals birthday object's month/day
        cout << "Happy Birthday!\n";    //if matched it's your birthday
    else
        cout << "It's not your birthday\n";  //if dates don't match it's not your birthday
    return 0;
}

//function definitions

void DayOfYear::output()
 {
  //displays the set month and day

    cout << "month = " << month
         << ", day = " << day << endl;

}

void DayOfYear::set(int new_month, int new_day) //setting
{
    month = new_month; //sets private variable month equal to argument new_month
    day = new_day;  //sets private variable day equal to argument new_day
    check_date(); //calling check_date to see if the month and date entered are valid
}

void DayOfYear::check_date( )
{
  //checking to see if month and date values are valid
    if ((month < 1) || (month > 12) || (day < 1) || (day > 31))
    {
        cout << "Illegal date. Aborting program.\n";
        exit(1);  //program exits if values are invalid
    }
}

int DayOfYear::get_month( )
{
    return month;    //retuns the private variable month
}

int DayOfYear::get_day( )
{
    return day;    //returns the private variable day
}
```



Example Explanation #

- First we make a class named `DayOfYear` and declare the `public` and `private` *variables*.
- `Public` variables include:
  - *Variable* `myVar`
  - The functions `output`, `set`, `get_month`, `get_day`
- `Private` variables include:
  - *Variables* `month`, `day`
  - *Fucntions* `check_date`

Let's take a look at all the *functions* one by one.

check_date()

- It checks whether the values of `month` and `day` are *valid*, if they are not, it gives an **error** and *aborts* the program.

set(int new_month, int new_day)

- In the example before when *member* variables were `public`, we set the `date` and `month` of an **object** by directly accessing these *member* variables in our `main` function using the *dot* operator.
- However, we can't set `private` variables directly in the `main` hence we use the `public` function `set` which can access these `private` variables. It takes the input arguments `new_month` and `new_day` and sets the values of `month` and `day` equal to them.
- It then calls the `check_date()` function to see if both *values* are valid.

get_month()

- *Returns* the value of `month` as it is a `private` can't be accessed directly in the `main` function.

get_day()

- *Returns* the value of `day` as it is a `private` can't be accessed directly in the `main` function.

output():

- Displays the `month` and `day`.

main()

- Declares **two** objects `today` and `birthday` for class `DayofYear`.
- *Calls* `set` function to *update* the values of `day` and `month` for both the objects.
- *Calls* the `output` function to display `today` date and `birthday` date.
- Lastly, it checks whether `today` date matches `birthday` date, if it does, it means it's their birthday.

Now in the next lesson, we will discuss *constructors* in classes in C++.