

7.6 Initialization and constructors

A good practice is to initialize all variables when declared. This section deals with initializing the data members of a class when a variable of the class type is declared.

Data member initialization (C++11)

Since C++11, a programmer can initialize data members in the class definition. Any variable declared of that class type will initially have those values.

Figure 7.6.1: A class definition with initialized data members.

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

class Restaurant {
public:
    void SetName(string restaurantName);
    void SetRating(int userRating);
    void Print();

private:
    string name = "NoName"; // NoName indicates name was not set
    int rating = -1;        // -1 indicates rating was not set
};

void Restaurant::SetName(string restaurantName) {
    name = restaurantName;
}

void Restaurant::SetRating(int userRating) {
    rating = userRating;
}

void Restaurant::Print() {
    cout << name << " -- " << rating << endl;
}

int main() {
    Restaurant favLunchPlace; // Initializes members with values in class
                             // definition

    favLunchPlace.Print();

    favLunchPlace.SetName("Central Deli");
    favLunchPlace.SetRating(4);

    favLunchPlace.Print();

    return 0;
}
```

```
NoName -- -1
Central Deli --
4
```

PARTICIPATION
ACTIVITY

7.6.1: Initialization.



Consider the example above.

- 1) When favLunchPlace is initially declared, what is the value of favLunchPlace's rating?

[Check](#)[Show answer](#)**Correct**

The Restaurant class definition has: `int rating = -1`. Thus, when favLunchPlace is declared as type Restaurant, that object's rating is initially -1. When favLunchPlace is first printed, the -1 was output.



- 2) After the call to SetRating(), what is the value of favLunchPlace's rating?

[Check](#)[Show answer](#)**Correct**

The call `favLunchPlace.SetRating(4)`; assigns rating with 4. The initial value of -1 is overwritten.

[Feedback?](#)

Constructors

C++ has a special class member function, a **constructor**, called *automatically* when a variable of that class type is declared, and which can initialize data members. A constructor callable without arguments is a **default constructor**, like the Restaurant constructor below.

A constructor has the same name as the class. A constructor function has no return type, not even void. Ex: `Restaurant::Restaurant() { ... }` defines a constructor for the Restaurant class.

If a class has no programmer-defined constructor, then the compiler *implicitly* defines a default constructor having no statements.

Figure 7.6.2: Adding a constructor member function to the Restaurant class.

```

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

class Restaurant {
public:
    Restaurant();
    void SetName(string restaurantName);
    void SetRating(int userRating);
    void Print();
private:
    string name;
    int rating;
};

Restaurant::Restaurant() { // Default constructor
    name = "NoName";        // Default name: NoName indicates name was not set
    rating = -1;            // Default rating: -1 indicates rating was not set
}

void Restaurant::SetName(string restaurantName) {
    name = restaurantName;
}

void Restaurant::SetRating(int userRating) {
    rating = userRating;
}

// Prints name and rating on one line
void Restaurant::Print() {
    cout << name << " -- " << rating << endl;
}

int main() {
    Restaurant favLunchPlace; // Automatically calls the default constructor

    favLunchPlace.Print();

    favLunchPlace.SetName("Central Deli");
    favLunchPlace.SetRating(4);
    favLunchPlace.Print();

    return 0;
}

```

```

NoName -- -1
Central Deli -- 4

```

[Feedback?](#)

PARTICIPATION ACTIVITY

7.6.2: Default constructors.



Assume a class named Seat.

- 1) A default constructor declaration in class Seat {

Correct

void shouldn't be there, just: Seat();



... } is:

```
class Seat {  
    ...  
    void Seat();  
}
```

- ☐ True
☒ False

- 2) A default constructor definition has this form:

```
Seat::Seat() {  
    ...  
}
```

- ☒ True
☐ False

Correct

The notation may look odd but tells the compiler that this is the constructor.



- 3) Not defining any constructor is essentially the same as defining a constructor with no statements.

- ☒ True
☐ False

Correct

Not finding any programmer-defined constructor, the compiler generates a constructor with no statements.



- 4) The following calls the default constructor once:

```
Seat mySeat;
```

- ☒ True
☐ False

Correct

The default constructor is called when an object is defined of the class type.



- 5) The following calls the default constructor once:

```
Seat seat1;  
Seat seat2;
```

- ☐ True
☒ False

Correct

The default constructor is called for each created Seat object, thus twice.



- 6) The following calls the default constructor 5 times:

```
vector<Seat> seats(5);
```

- ☒ True
☐ False

Correct

The vector definition creates 5 elements, each being a Seat object. Thus, the Seat constructor is called 5 times.



Note: Since C++11, data members can be initialized in the class definition as in `int price = -1;`, which is usually preferred over using a constructor. However, sometimes initializations are more complicated, in which case a constructor is needed.

Exploring further:

- [Constructors](#) from msdn.microsoft.com

**CHALLENGE
ACTIVITY**

7.6.1: Basic constructor definition.



Define a constructor as indicated. Sample output for below program:

Year: 0, VIN: -1

Year: 2009, VIN: 444555666

```
1 #include <iostream>
2 using namespace std;
3
4 class CarRecord {
5     public:
6         void    SetYearMade(int originalYear);
7         void    SetVehicleIdNum(int vehIdNum);
8         void    Print() const;
9         CarRecord();
10    private:
11        int     yearMade;
12        int     vehicleIdNum;
13 };
14
15 // FIXME: Write constructor, initialize year to 0, vehicle ID num to -1.
16
17 /* Your solution goes here */
18 CarRecord::CarRecord(){
19     yearMade = 0;
20     vehicleIdNum = -1;
21 }
22
23 void CarRecord::SetYearMade(int originalYear) {
```

Run

All tests passed

Printing immediately after defining car object

Your output

Year: 0, VIN: -1

✓ Printing after setting year to 2009, VIN to 444555666.

Your output

```
Year: 2009, VIN: 444555666
```

[Feedback?](#)