Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Putting it all together for polymorphism and virtual functions

Recap

IIT Bombay

- Polymorphism in C++ programming
- Virtual destructor
- Abstract class

Overview of This Lecture

IIT Bombay

- An example program on polymorphism

Acknowledgement

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

Problem Statement

IIT Bombay

We want to write a program to:

- Take input (a verb) from the user
  - Print its past tense

Verbs

IIT Bombay

Regular

Play → Played
Walk → Walked
Look → Looked
Watch → Watched

Irregular

Be → **Was**
Speak → **Spoke**
Eat → **Ate**

Possible Solution

Suffix '**ed**' whenever required

Store every verb and its past tense

Solution

IIT Bombay

- Represents all verbs
- Base class of class ‘regular’ and ‘irregular’
- Definition of verb contains information common to all verbs
- Data member root contains the verb itself (supplied by the user as input)

```
class verb{
  protected:
    string root;
  public:
    string getRoot() {
       return root;
    }
    virtual string past_tense() = 0;
};
```

Solution

IIT Bombay

- Stores regular verbs in ‘root’ (inherited from class ‘verb’)
- Member function ‘past_tense()’ returns the past tense of the verb by suffixing ‘ed’ to the verb (stored in root).

```
class regular : public verb{
  public:
    regular(string rt) {
       root = rt;
    }
    string past_tense() {
       return root + "ed";
    }
};
```

Solution

IIT Bombay

- Stores irregular verbs in ‘root’ (inherited from class ‘verb’)
- Stores the past tense of such verbs in private member ‘pt’,
  - passed as a constructor argument.
- Member function ‘past_tense()’ returns the past tense of the verb, which is stored in ‘pt’.

```
class irregular : public verb{
    string pt;
  public:
    irregular(string rt, string p) {
      root = rt;
      pt = p;
    }
    string past_tense() {
      return pt;
    }
};
```

Solution

IIT Bombay

```
int main() {

  verb *v1;
  regular r[2] = { regular("play"), regular("watch") };
  irregular ir[3] = { irregular("is","was"), irregular("go","went"), irregular("speak","spoke") };

  string query;
  do {
      ...
      ...
  } while (true) ;

  return 0;
}
```

Solution

```
do{
    cout << "Enter the verb to find: " ;
    cin >> query;
    bool found = false;
    //Loop for the list of regular verbs
    for (int i =0; i < 2; i++){
        v1 = &r[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true;  //Verb found in this list
            break;
        } // End of if
    } // End of for
    if (found == true) //Verb found in the list
        continue; //Ask user for another verb
    //Loop for the list of irregular verbs
    for (int i =0; i < 3; i++) {
        v1 = &ir[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true; //Verb found in this list
            break;
        } //End of if
    } //End of for
    if (found == false) //Verb not found in both lists
        cout << "Verb not found" << endl;
} while(true);  //End of do..while
```

Solution

```
do{
    cout << "Enter the verb to find: " ;
    cin >> query;
    bool found = false;
    //Loop for the list of regular verbs
    for (int i =0; i < 2; i++){
        v1 = &r[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true;  //Verb found in this list
            break;
        } // End of if
    } // End of for
    if (found == true) //Verb found in the list
        continue; //Ask user for another verb
    //Loop for the list of irregular verbs
    for (int i =0; i < 3; i++) {
        v1 = &ir[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true; //Verb found in this list
            break;
        } //End of if
    } //End of for
    if (found == false) //Verb not found in both lists
        cout << "Verb not found" << endl;
} while(true);  //End of do..while
```

Solution

```
do{
    cout << "Enter the verb to find: " ;
    cin >> query;
    bool found = false;
    //Loop for the list of regular verbs
    for (int i =0; i < 2; i++){
        v1 = &r[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true;  //Verb found in this list
            break;
        } // End of if
    } // End of for
    if (found == true) //Verb found in the list
        continue; //Ask user for another verb
    //Loop for the list of irregular verbs
    for (int i =0; i < 3; i++) {
        v1 = &ir[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true; //Verb found in this list
            break;
        } //End of if
    } //End of for
    if (found == false) //Verb not found in both lists
        cout << "Verb not found" << endl;
    } while(true);  //End of do..while
```

Solution

```
do{
        cout << "Enter the verb to find: " ;
        cin >> query;
        bool found = false;
        //Loop for the list of regular verbs
        for (int i =0; i < 2; i++){
            v1 = &r[i];
            //(to compare input of user with our list)
            if (v1->getRoot() == query){
                cout << "Past tense of “;
                cout << v1->getRoot() << " is “;
                cout << v1->past_tense() << endl;
                found = true;  //Verb found in this list
                break;
            } // End of if
        } // End of for
        if (found == true) //Verb found in the list
             continue; //Ask user for another verb
        //Loop for the list of irregular verbs
        for (int i =0; i < 3; i++) {
            v1 = &ir[i];
            //(to compare input of user with our list)
            if (v1->getRoot() == query){
                cout << "Past tense of “;
                cout << v1->getRoot() << " is “;
                cout << v1->past_tense() << endl;
                found = true; //Verb found in this list
                break;
            } //End of if
        } //End of for
        if (found == false) //Verb not found in both lists
            cout << "Verb not found" << endl;
    }  while(true);  //End of do..while
```

Solution

```
do{
    cout << "Enter the verb to find: " ;
    cin >> query;
    bool found = false;
    //Loop for the list of regular verbs
    for (int i =0; i < 2; i++){
        v1 = &r[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true;  //Verb found in this list
            break;
        } // End of if
    } // End of for
    if (found == true) //Verb found in the list
        continue; //Ask user for another verb
    //Loop for the list of irregular verbs
    for (int i =0; i < 3; i++) {
        v1 = &ir[i];
        //(to compare input of user with our list)
        if (v1->getRoot() == query){
            cout << "Past tense of “;
            cout << v1->getRoot() << " is “;
            cout << v1->past_tense() << endl;
            found = true; //Verb found in this list
            break;
        } //End of if
    } //End of for
    if (found == false) //Verb not found in both lists
        cout << "Verb not found" << endl;
    }  while(true);  //End of do..while
```

Solution

```
do{
      cout << "Enter the verb to find: " ;
      cin >> query;
      bool found = false;
      //Loop for the list of regular verbs
      for (int i =0; i < 2; i++){
          v1 = &r[i];
          //(to compare input of user with our list)
          if (v1->getRoot() == query){
               cout << "Past tense of “;
               cout << v1->getRoot() << " is “;
               cout << v1->past_tense() << endl;
               found = true;  //Verb found in this list
               break;
          } // End of if
      } // End of for
      if (found == true) //Verb found in the list
           continue; //Ask user for another verb
```

```
//Loop for the list of irregular verbs
    for (int i =0; i < 3; i++) {
          v1 = &ir[i];
          //(to compare input of user with our list)
          if (v1->getRoot() == query){
               cout << "Past tense of “;
               cout << v1->getRoot() << " is “;
               cout << v1->past_tense() << endl;
               found = true; //Verb found in this list
               break;
          } //End of if
    } //End of for
    if (found == false) //Verb not found in both lists
         cout << "Verb not found" << endl;
    }  while(true);  //End of do..while
```

Sample Output

IIT Bombay

```
Enter the verb to find: play
Past tense of play is played
Enter the verb to find: go
Past tense of go is went
Enter the verb to find: speak
Past tense of speak is spoke
Enter the verb to find: have
Verb not found
Enter the verb to find: watch
Past tense of watch is watched
Enter the verb to find: ^C
```

Summary

- Written a complete program that prints the past tense of regular and irregular verbs
  - Shown polymorphism using virtual functions
  - Used pure virtual functions, abstract class