Question and ChoiceQuestion

Goal

- Write programs to post questions and answer them, check the answer is correct or not.
- Questions can be one of the following.
    - ❑Open-end question, the answer needs to be an exact match.
    - ❑Multiple choice questions, display a question with several options, choose from one of those options. Answer needs to match the label of the correct answer.
    - ❑Numerical question, where the answers are in numbers can be approximate. For example, when 1.0 divided by 3, the answer can be 0.3333 or 0.333.

Question class

- Think about an open-end question, which includes a problem text and its answer.
- For example, text of question is “When was the USA founded?”, its answer is “1776”.
  - We use a string to save answer, since not all questions can be answered by numbers. For example, “Who invented C++?”.
  - Any number can be converted to a string using to_string method of string class. See https://www.cplusplus.com/reference/string/to_string/.
- Operations includes setters and getters for data members. In addition, need a method to check whether a reply is correct or not. Call this method checkAnswer.

Question.hpp

```
#ifndef QUESTION_H
#define QUESTION_H
#include <string>
using namespace std;
class Question {
public:
   Question();
   string getText() const; //Return problem text.
   string getAnswer() const; //Return answer.
   void setText(string text); //Update data member text by input parameter text.
   void setAnswer(string answer);
   bool checkAnswer(string reply) const;
   virtual void display() const;
private:
   string text; //problem text
   string answer; //answer to question
};
#endif
```

- Key word virtual means function display will be overridden by the same-header function in subclass.
- Display an (open-end) question is different from displaying a choice question or a true/false question.
- Keyword virtual is needed for polymorphism, which enables codes defined for super class can apply to ANY object of its subclasses.

ChoiceQuestion

- Example of ChoiceQuestion

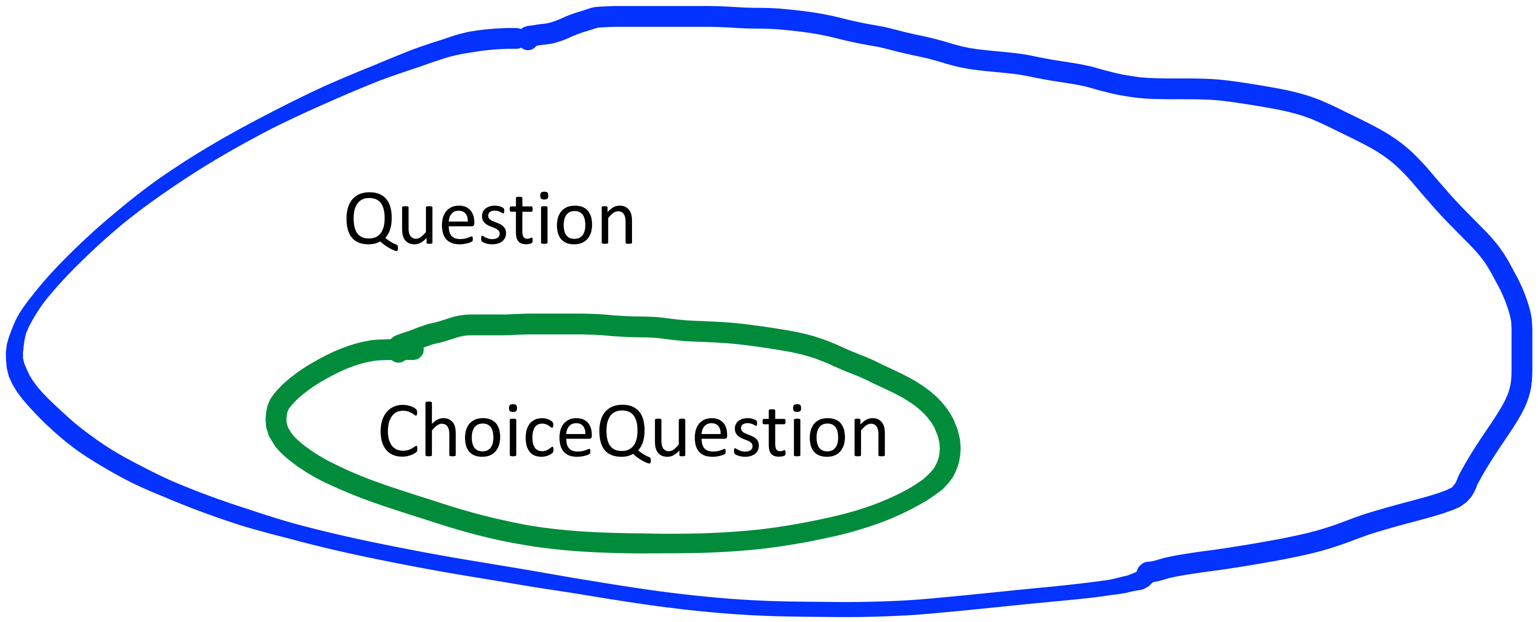
  Question: What is 1 + 1?

  1. 0
  2. 1
  3. 2
  4. 3

- When answer a question with choice, we choose the option label besides the correct answer. In the above example, choose Option 3.
  - Unlike an open-end question, in a choice question, the answer is the option chosen.
- So, besides text and answer to (open-end) questions, need to list choices associated with a choice question.

Question vs. ChoiceQuestion

- Every choice question is-a question.
- A choice question is a question with options provided.



Represent choices of a choice question

- Use a vector of strings.
- Add a choice and include a bool value to represent whether this choice is a correct answer or not.

  Question: What is 1 + 1?
  1. 0
  2. 1
  3. 2
  4. 3
- The content of first choice is 0, which is not an answer to 1 + 1. So this choice is false.
  - A choice needs to be represented as a string to be consistent with type of answer. Furthermore, not every choice can be represented by a number.
- The third choice is 2, which is an answer to this problem, so this choice is true.

Method addChoice in ChoiceQuestion

void addChoice(string option, bool isCorrect)

- When we add an option, push it back to vector of strings called choices.
- Vector choices is empty in the very beginning. The order of pushing back implies the labelling of this option.
- First add string “0”. Note that each answer candidate is stored as a string.
  - After push_back operation of vector choices, the vector has 1 element, which is reflected in its current size.
  - The size is consistent with the labelling of the current option, which is 1 now.
- Then we add string “1”.
  - After push_back operation of vector choices, the vector has 2 elements, which is reflected in its size.
  - The current size is 2, the order of this option is 2.

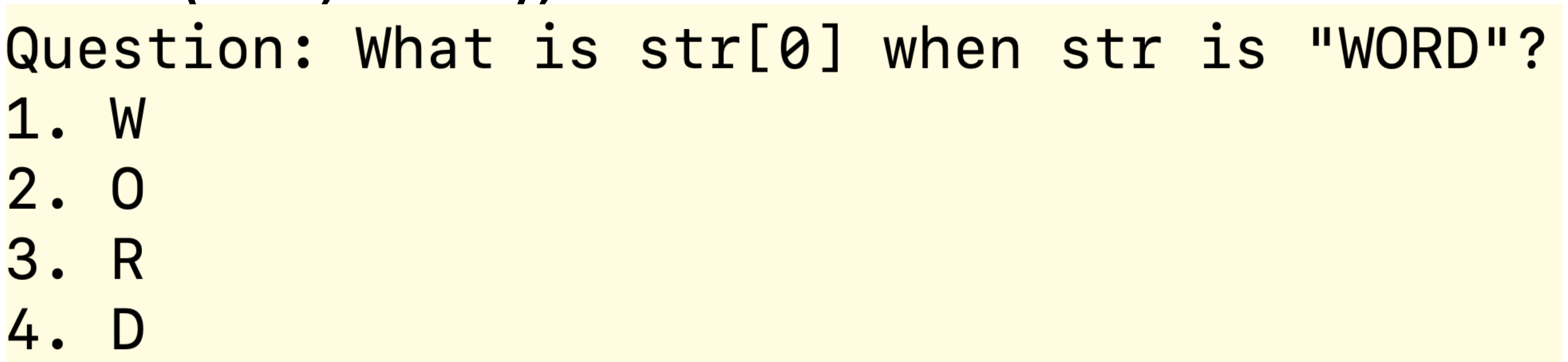
ChoiceQuestion example

```
question: What is 1 + 1?
1. 0
2. 1
3. 2
4. 3
```

```
ChoiceQuestion onePlusOne; //set text and answer to be empty strings
onePlusOne.setText("What is 1 + 1?"); //update question text
onePlusOne.addChoice("0", false); //push "0" to vector choices
onePlusOne.addChoice("1", false); //push "1" to vector choices
onePlusOne.addChoice("2", true); //push "2" to vector choices,
    //set "3" to be answer. This is the third item of the choices.
    //3 is the label of this option. Each answer is a string.
onePlusOne.addChoice("3", false); //push ”3" to vector choices
```

ChoiceQuestion example: II

```
ChoiceQuestion stringIndex;
stringIndex.setText("What is str[0] when str is \"WORD\"?");
stringIndex.addChoice("W", true);
stringIndex.addChoice("O", false);
stringIndex.addChoice("R", false);
stringIndex.addChoice("D", false);
```



```
Question: What is str[0] when str is "WORD"?
1. W
2. O
3. R
4. D
```

Common mistakes in inheritance

- Forget to add : public superclassName after class subclass declaration.
- Re-define data members from superclass.
- Re-define every method from superclass.
    - If so, why do we need to use inheritance?

Question vs Choice Question

- A question has text and answer.
- A choice question is a question with options.
    - When displaying a choice question, besides text, options are displayed.
- Question is super class, choice question is subclass.

Question vs Numerical Question

- A numerical question is a question whose answer is a number.
- Shall we add a double-type data member to represent answer in a numerical question?
- We can, but it is not a good idea – a data member need its own getter and setter.
- NumericalQuestion inherits data members from Question class.
- Just override (re-define) checkAnswer method inherited from Question class to do an approximate match.

Define a subclass

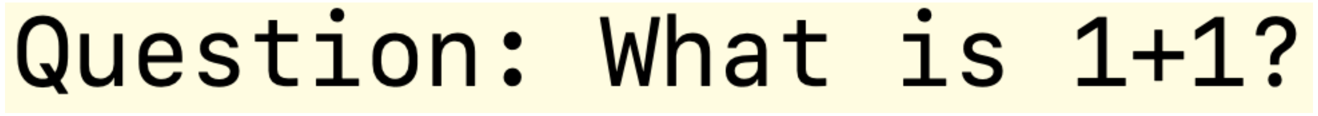
- In declaration, class subclassName : public superclassName
- In subclass, only define data members not in superclass.
- In subclass, define constructors for subclass – constructors in superclass are not inherited, but constructors in superclass can be involved in subclass.
- In subclass, only define setters and getters for data members unique to subclass.
- In subclass, only override those methods inherited from superclass but behave differently in subclass. For those method declaration, add virtual in superclass for polymorphism.
- For method inherited from superclass but behave the same in subclass, do not need to do anything – just inherit them without a single line modification.

Polymorphism

- Goal: define a method or function for a superclass and hopefully the method can apply to ALL subclasses of that superclass.
- For example, define reply function, taking an object of Question as formal parameter, do the following.
    - ❑ Display question
    - ❑ Get answer from user
    - ❑ Check whether answer is correct or not.

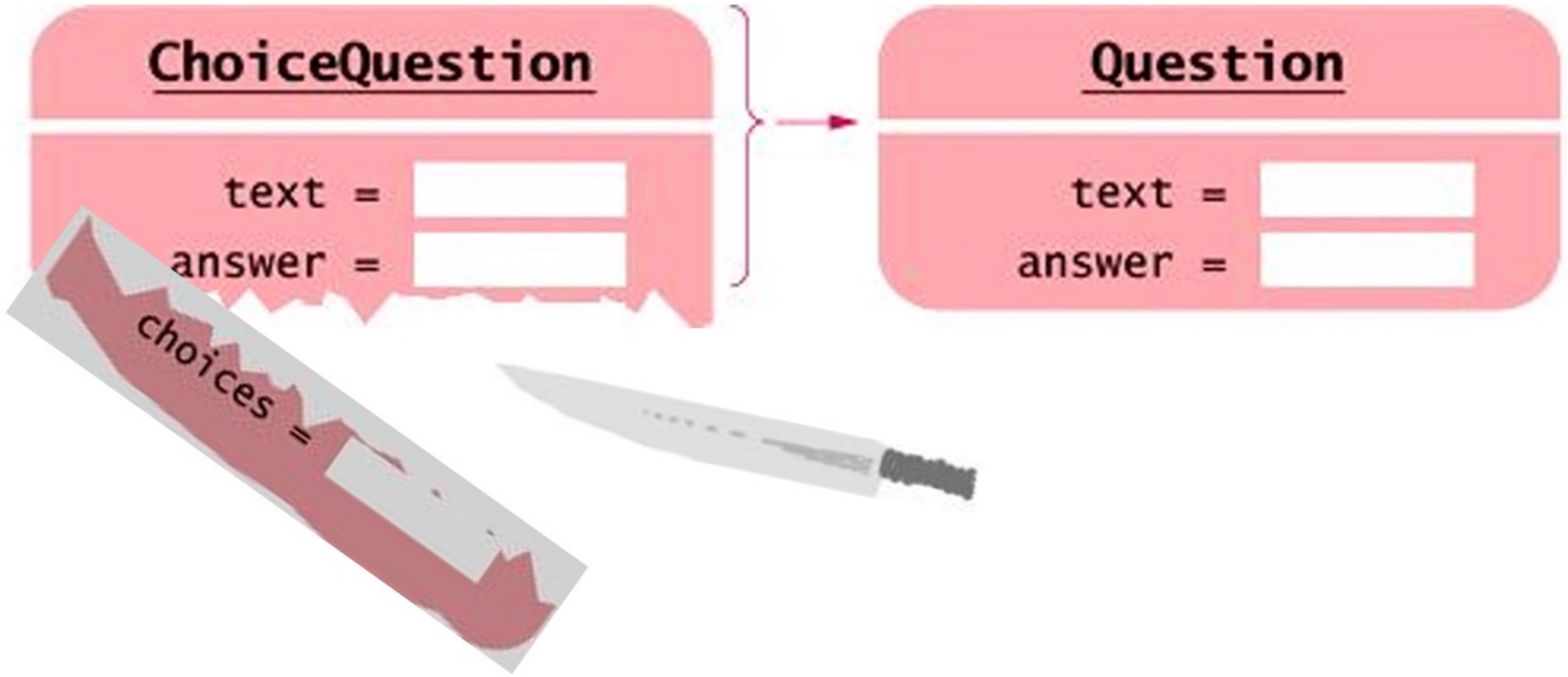
Polymorphism: will the following code work?

```
Question ques[] = {usaFounded, onePlusOne, oneDividedBy3};
int size = sizeof(ques) / sizeof(ques[0]);
for (int i = 0; i < size; i++) {
    ques[i].display();
    getline(cin, reply);
    cout << boolalpha << ques[i].checkAnswer(reply) << endl;
}
```



```
Question: What is 1+1?
```

Slicing problem: no options displayed for ChoiceQuestion



Reason for Slicing Problem

- The array ques holds Questions.

Question ques[] = {usaFounded, onePlusOne, oneDividedBy3};

- The compiler realizes that a ChoiceQuestion is a special case of a Question. Thus it permits to set ques[1] by onePlusOne;
- However, a ChoiceQuestion object has 3 data members, whereas a Question has just 2.
- There is no room to store the derived-class data in the array.
- That data simply gets sliced away when you assign a derived-class object to a base-class variable.

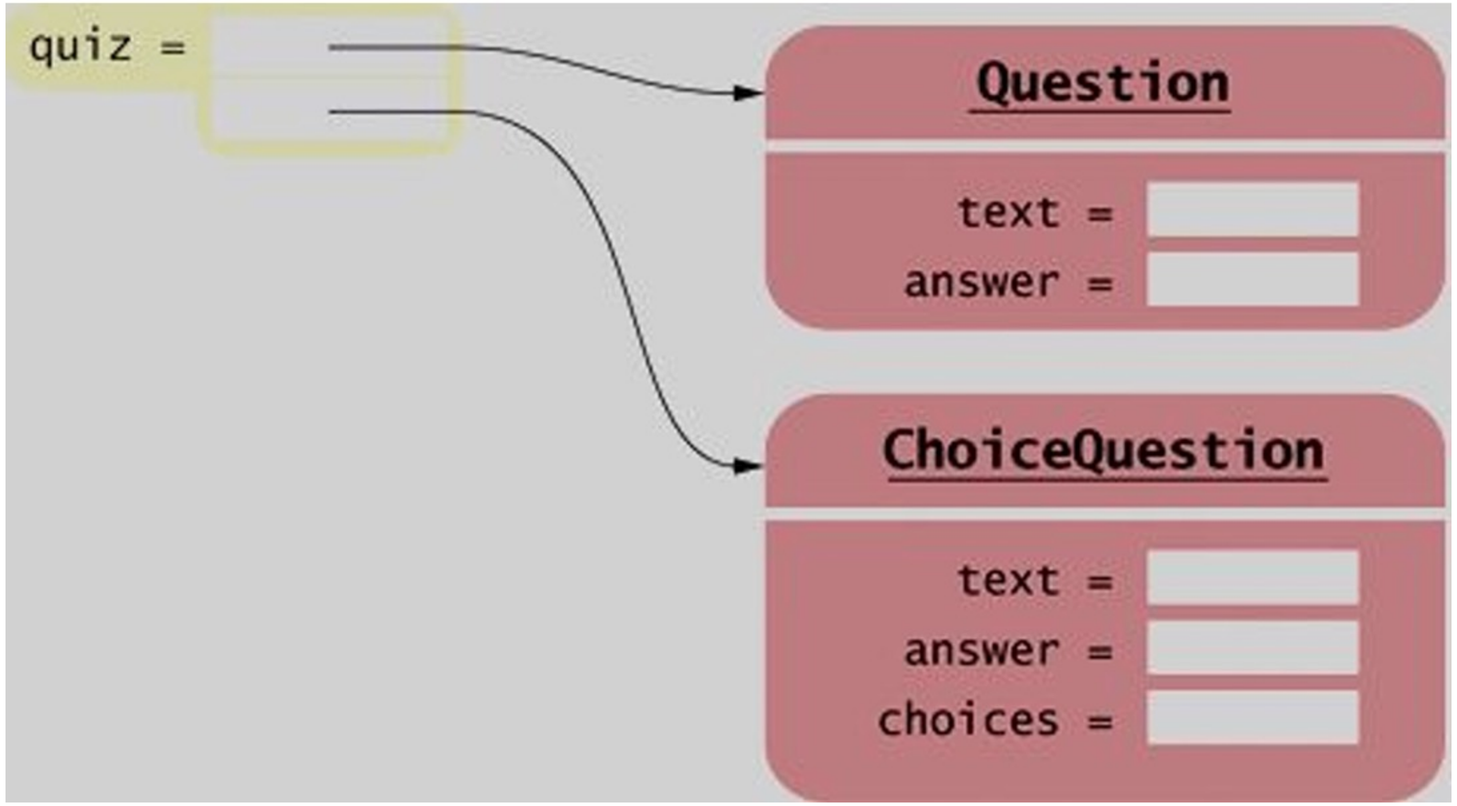
Handle Slicing Problem

Pointer to the rescue!

```
Question* quiz[] = {&usaFounded, &onePlusOne, &oneDividedBy3};
int size = sizeof(quiz) / sizeof(quiz[0]);
for (int i = 0; i < size; i++) {
    quiz[i]->display();
    getline(cin, reply);
    cout << boolalpha << quiz[i]->checkAnswer(reply) << endl;
}
```

Handle Slicing Problem: II

- Pointer to the rescue!



Define a function that works for all questions

```
void reply(Question* q) {
        q->display();
        string usrReply;
        getline(cin, usrReply);
        cout << boolalpha << q->checkAnswer(usrReply) << endl << endl;
}
```

Polymorphism

Each object knows on its own

how to carry out its specific version of these general tasks

“Display the question”   (my way)   and

“Check a response”   (my way).

Complete code in onlinegdb

- Complete code for Question, ChoiceQuestion, and NumericalQuestion classes and test them