Теория и технология программирования

Основы программирования на языках C и C++

Лекция 10. Инкапсуляция

Глухих Михаил Игоревич, к.т.н., доц.
mailto: glukhikh@mail.ru

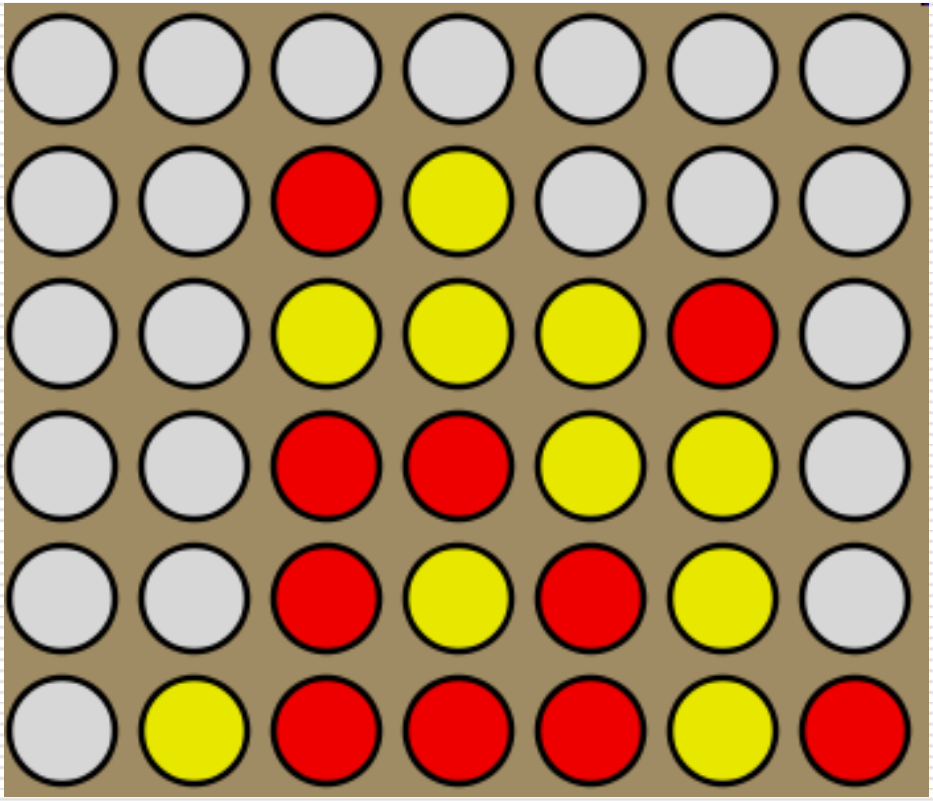
Классы и структуры в C++

- Обеспечивают механизм создания собственных типов и определения различных действий над ними
- Обычно используются для описания различных понятий, фигурирующих в решаемой задаче

Объектно-ориентированное проектирование

- Определить, какие понятия фигурируют в решаемой задаче и как они связаны друг с другом
- Реализовать нетривиальные понятия в виде классов (объектов). Каждому классу соответствует модуль на языке C++
- Решить основную задачу с использованием данных понятий и действий над ними

Пример - игра «Четыре в ряд»



Правила игры

- Используется прямоугольное поле шириной в 7 клеток и высотой в 6
- Красный и желтый игроки ходят по очереди
- Ход заключается в добавлении фишки своего цвета в одну из колонок поля, при этом заполняется самая нижняя из свободных клеток этой колонки
- Побеждает игрок, составивший четыре фишки своего цвета в ряд по горизонтали, вертикали или диагонали
- Если поле заполнено и никто не выиграл - объявляется ничья

Задача

- Реализовать игру «Четыре в ряд» в варианте «Человек против человека» в консольном режиме

Вопросы проектирования

- Какие понятия фигурируют в задаче?
- Какие действия можно над ними совершать?
- Чем описываются данные понятия?

Игровое поле – методы

- Действия
  - Создание/очистка
  - Выполнение хода
- Доступ
  - К ячейкам
  - К очередности хода
  - Определение победителя/момента окончания игры
- Представление
  - Печать поля
  - Печать состояния

Игровое поле – методы

```
class Field
{
public:
    Field(bool isRedFirst);
    void clear(bool isRedFirst);
    bool makeTurn(int column);
    bool isWon(bool red) const;
    bool isOver() const;
    Cell getCell(int i, int j) const;
    bool isRedTurnNow() const;
    void print() const;
    void printResult() const;
};
```

Содержимое ячейки

- Ячейка может быть в трех состояниях – пустая, красная, желтая
- Содержимое ячейки может быть задано целым числом – например, 0 для пустой ячейки, 1 для красной фишки, или 2 для желтой фишки
- Второй, более правильный вариант – использование **перечислений**

Игровое поле – ячейка

```
// Перечисление
enum Cell
{
   EMPTY,
   RED,
   YELLOW
};
// Переменная типа «перечисление» может
// равняться одному из четырех
// перечисленных значений
Cell cell=RED;
```

Перечисления

- В памяти компьютера хранятся как целые числа
- По умолчанию, первому элементу перечисления соответствует 0, второму 1 и так далее
- Разрешается указать целые значения явно:

```
enum Cell
{
    EMPTY=0,
    RED=2,
    YELLOW=8
};
```

- Возможно преобразование из перечисления в целое и обратно

```
Cell cell=(Cell)2;
int code=(int)cell;
```

Преимущества перечислений

- В целую переменную можно записать любое целое значение, в переменную типа «перечисление» - только одно из заданных значений (то есть, устраняется возможная некорректность)
- Кроме этого, записи с перечислениями лучше читаются, сравните:

```
Cell cell=RED;
int cell=1;
```

Игровое поле – данные

- Ячейки (двумерный массив)
- Кто ходит
- Кто победил

Игровое поле – данные

```
class Field
{
    // Двумерный массив для
    // хранения игрового поля
    Cell cells[FIELD_WIDTH][FIELD_HEIGHT];
    // Очередь хода
    bool isRedTurn;
    // Кто на данный момент выиграл
    Cell winner;
    // …
};
```

Определение класса целиком

- Пример Four, файл Field.h

Конструктор и очистка

```
Field::Field(bool isRedFirst)
{
   clear(isRedFirst);
}

void Field::clear(bool isRedFirst)
{
   isRedTurn = isRedFirst;
   winner = EMPTY;
   for (int i=0; i<FIELD_WIDTH; i++)
      for (int j=0; j<FIELD_HEIGHT; j++)
         cells[i][j] = EMPTY;
}
```

Выполнение хода

```
bool Field::makeTurn(int column)
{
   if (winner != EMPTY ||
       column < 1 || column > FIELD_WIDTH)
      return false;
   int i=column-1;
   for (int j=0; j<FIELD_HEIGHT; j++)
      if (cells[i][j]==EMPTY)
      {
         cells[i][j] = isRedTurn ? RED : YELLOW;
         checkWinner(); // Победа?
         isRedTurn = !isRedTurn;
         return true;
      }
   return false;
}
```

Определение победителя

- Необходимо найти ряд из четырех одинаковых клеток - по горизонтали, вертикали или диагонали
- Как это лучше сделать?

Определение победителя

```
const int DIR_NUMBER = 4;
const int di[] = { 1, 0, 1, 1 };
const int dj[] = { 0, 1, -1, 1 };
const int WIN_LENGTH = 4;
void Field::checkWinner()
{
   for (int i=0; i<FIELD_WIDTH; i++)
      for (int j=0; j<FIELD_HEIGHT; j++)
      {
         Cell start = cells[i][j];
         if (start==EMPTY) continue;
         for (int dir=0; dir<DIR_NUMBER; dir++)
         { ... }
      }
}
```

Определение победителя

```
int length=0, iline = i, jline = j;
while (++length < WIN_LENGTH)
{
   iline += di[dir];
   jline += dj[dir];
   if (iline < 0 || iline >= FIELD_WIDTH ||
       jline < 0 || jline >= FIELD_HEIGHT)
      break;
   if (cells[iline][jline]!=start) break;
}
if (length==WIN_LENGTH)
{
   winner = start;
   return;
}
```

Контроль окончания игры

```
bool Field::isOver() const
{
   if (winner != EMPTY)
      return true;
   for (int i=0; i<FIELD_WIDTH; i++)
      for (int j=0; j<FIELD_HEIGHT; j++)
         // Если хоть одна ячейка свободна,
         // игра не окончена
         if (cells[i][j]==EMPTY)
            return false;
   // Все ячейки заняты
   return true;
}
```

Другие методы

- Пример Four, файл Field.cpp

Главная функция

```
int main(void)
{
   setlocale(LC_ALL, "Russian");
   Field field(true);
   while (!field.isOver())
   {
      field.print();
      cout<<"Ходит "<<
         (field.isRedTurnNow() ? "красный" : "желтый")<<
         " игрок, введите ход (1-7)"<<endl;
      int column; cin>>column;
      field.makeTurn(column);
   }
   field.printResult();
   return 0;
}
```

Демонстрация

- См.

Как тестировать?

Как тестировать?

- Долго играть
  - по-хорошему, это надо делать каждый раз, когда что-то изменяем
- Написать одну или несколько функций, которые будут имитировать игру
- Написать одну универсальную функцию

Итоги

- Рассмотрено
  - Порядок объектно-ориентированного проектирования
  - Методы и данные на примере игрового поля
  - Перечисления
- Далее
  - Область действия и время жизни
  - Лексический анализ