Гараев И.А.[1], Нарбаев И.Б.[1], Петрова Н.К.[2], Петрова А.С.[2©]
Студент[1], доцент, к.ф.-м.н. [2], КГЭУ

ПРИНЦИПЫ ПОСТРОЕНИЯ ТЕСТОВЫХ ПРИМЕРОВ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ПРОЕКТА

Аннотация

В статье рассматриваются принципы построения тестовых примеров с точки зрения их применения для формирования алгоритма решения поставленной задачи, проверки полноты алгоритма и контроля получаемого в программе решения. Корректная разработка тестовых примеров – важнейший этап в разработке любого программного продукта. Он определяет скорость разработки и качество получаемого решения. Однако в учебной практике вузов разработке этого важного элемента программирования уделяется мало внимания. Особые сложности возникают при формировании алгоритмов не вычислительных задач. В статье этот вопрос рассмотрен для случая программирования компьютерной игры - «Угадай число». Пример программ написан на C++.

Ключевые слова: контрольный пример, полнота задачи, программирование игр.

Введение.

В рамках учебного курса «Алгоритмизация и программирование» особую сложность вызывают не только формирование алгоритма решения задачи, но и вопросы отладки и тестирования, разработанной программы. Не имеющие опыта студенты редко беспокоятся о контроле своего решения: компьютер выдал результат – и хорошо. А соответствует ли он реальному решению – вопрос часто и не возникает.

Но умение грамотно разработать контрольные примеры очень важно при построении любого, как сложного, так и простого, программного продукта. Проработанный деталях тестовый пример позволяет уловить те закономерности, которые позволят сформировать алгоритм решения. Пи этом, важно именно на тестовом примере отработать соответствие полноте решения. Это означает, что контрольный пример позволяет проработать все возможные ситуации с входящими данными и предусмотреть исходы решения, особенно, если при каких-то начальных значения возможны в программе разрывы, не санкционированные прерывания, остановки.

В случае вычислительных задач ситуацию с полнотой решения проще решить на математическом уровне: рассмотреть области определения функции, области бифуркации решения при изменении параметров и т.п. Однако решение таких задач является, как правило , прерогативой инженерных и научно-технических направлений.

В современных условиях область применения программных средств расширилась существенно: от научных до бытовых проблем. Сильным стимулом в развитии и программного, и аппаратного обеспечения являются задачи по построению компьютерных игр. Игры являются в тоже время и хорошим объектом для научения студентов основам простейшего компьютерного моделирования. Но именно с подобного рода задачами, у студентов, как показывает опыт, возникает немало проблем по разработке алгоритма и тестированию получаемого результата.

В представленной статье мы поделимся опытом разработки простейшей игры «Угадай число», при анализе которой существенное внимание уделим разработке контрольного примера.

В реальных ситуациях игра «Угадай число» интегрирована во множество проектов. В большинстве случаев используются аналоги этого процесса, где числа заменяются

© Гараев И.А., Нарбаев И.Б., Петрова Н.К., Петрова А.С., 2020

какими-либо иными объектами, в зависимости от проекта и его целей, но суть игры остается прежней. Существует большая масса игр, в которые интегрирована данная игра, где она подразумевает под собой некую мини-игру, где человек может попытать свою удачу. Еще один пример, где можно встретить эту игру - это экзамены в том или ином учебном заведении. Перед студентом стоит задача выбрать билет, желательно тот, который он знает лучше остальных. И выбор данного билета и является победой в этой условной игре [1].

Постановка задачи для разработки игры.

Продумаем этапы построения программы, реализующей простейшую игру с компьютером. Суть игры в следующем. Игроку сообщается о начале игры и её правилах. Количество попыток в игре представлено в виде «жизней», которые генерируются случайным образом в заданном диапазоне значений, например, от 5 до 10. Целью игрока является угадать число, «загаданное» компьютером. После каждой неудачной попытки игроку сообщается: выбранное число больше или меньше «загаданного». Каждая попытка затрачивает одну жизнь. Игра завершается победой в случае угадывания. Если кол-во попыток заканчивается или игрок больше не желает играть, процесс игры останавливается.

Сформулированная таким образом игра требует формализации, по которой будет переводиться на язык, понятный компьютеру. Помочь в этом может проработка контрольного примера.

Рассмотрим несколько вариантов программируемого процесса.

Вариант 1: Игрок угадывает число.

1) Компьютер загадал число 21 (в диапазоне от 10 до 50).

Кол-во жизней: 6

2) Игрок вводит число 11 (оно меньше 21, но удовлетворяет условиям продолжения игры).

3) Компьютер говорит: меньше и снимает одну жизнь. Предлагает следующую попытку. (кол-во жизней: 5)

4) Игрок вводит число 30.

5) Компьютер говорит: больше и снимает одну жизнь. Предлагает следующую попытку. (кол-во жизней: 4)

6) Игрок вводит число 21.

7) Компьютер оповещает о том, что Игрок победил, завершает игру!

Вариант 2: Игрок не угадывает число при выделенном количестве жизней.

Чтобы сократить кол-во операция присвоим кол-во жизней 3.

1) Компьютер загадал число 34 (в диапазоне от 10 до 50).

Кол-во жизней: 3

2) Игрок вводит число 49 (оно больше 34, но удовлетворяет условиям продолжения игры).

3) Компьютер говорит: больше и снимает одну жизнь. Кол-во жизней: 2.

4) Игрок вводит число 40 (оно больше 34, но удовлетворяет условиям продолжения игры).

5) Компьютер говорит: больше и снимает еще одну жизнь. Кол-во жизней: 1.

6) Игрок вводит число 35 (оно больше 34, но удовлетворяет условиям продолжения игры).

7) Компьютер сообщает о том, что жизней больше нет и игрок проиграл.

Вариант 3: Игрок хочет прервать игру.

1) Компьютер загадал число 12 (в диапазоне от 10 до 50).

Кол-во жизней: 8

2) Игрок вводит число 100. Оно не входит диапазон значений, а по правилам игры это означает прекращение игры.

3) Компьютер печатает о том, что игра закончилась по желанию игрока.

В этих трёх вариантах мы рассмотрели все возможные исходы работы программы. Завершение процесса предусмотрено вводи числа, выходящего за диапазон возможных значений. При этом интерфейс программы должен известить игрока обо всех нюансах игры, в том числе и о возможностях её досрочного прерывания.

Анализ контрольных примеров позволяет сформировать нам и алгоритм решения задачи. Для неё потребуются следующие операции.

- Генерация случайных чисел в заданном диапазоне для «загаданного» числа, которое потребуется угадать Игроку.
- Использование обратного счётчика для контроля количества жизней (когда кол-во жизней достигнет отметку «ноль», игра считается завершенной).
- Алгоритм ветвления для сравнения введенного Игроком числа с «загаданным» и выводом соответствующего сообщения на консоль.

Для последующего контроля за работой программы распишем стратегию проведения процесса игры, сопровождая каждый шаг числовыми расчётами. решения этой задачи может быть описана следующим образом:

Вводится диапазон чисел, в котором случайным образом будет задано число, которое потребуется угадать: 10 50.

Пусть компьютером было сгенерировано число 29.

И кол-во жизней L= 7.

Начинается цикл пока (L == true)

1) Игрок задаёт число 20.

20 == 29? Нет! Значит вызываем функцию-подсказку: 20 < 29? Да! Сообщаем об этом игроку.

Отнимаем одну жизнь.

Текущее состояние жизней: 6

2) Игрок задаёт число 30.

30 == 29? Нет! Значит вызываем функцию-подсказку: 20 < 29? Нет! Сообщаем об этом игроку.

Отнимаем одну жизнь.

Текущее состояние жизней: 5.

3) Игрок задаёт число 29.

29 == 29? Да! Игрок победил.

На этом примере мы получили не только числовой контроль для программы, но, фактически, можем видеть словесный алгоритм решения задачи.

Для оптимизации процесса разработки, естественно применение структурного подхода. Анализ приведённого фрагмента процесса игры позволяет нам определить те алгоритмические элементы, которые могут быть вынесены в отдельные модули [2,3].

Алгоритмизация

Алгоритм главной программы, основанный на анализе вышеописанного примера, будет иметь следующую струкутру:

1) Вводим интервал, внутри которого задается число.
2) Генерируем кол-во жизней (от 5 до 10)
3) Выводим на консоль правила игры
4) Игрок вводит число.
5) Программа переходит на проверку условия выхода: а) игрок угадал б) игрок больше не желает играть.
6) Если условия выхода не выполнены, компьютер подсказывает: введённое число больше или меньше заданного - и снимает одну жизнь.
7) Перед каждым запросом на ввод числа программа информирует о текущем состоянии жизней

8) Если попытки закончились, или Игрок угадал, или он прервал игру, выдаётся соответствующее сообщение.

Из анализа алгоритма можно выделить повторяющиеся фрагменты программы и логически целостные вычислительные блоки. Эти блоки реализуются в виде функций:

Функция **set_Min_And_Max** позволяет ввести диапазон генерации случайных чисел.

Функция **rand_Value** генерирует случайные числа в некотором диапазоне.

Функция **calculate_Value** является главной по вычислительной части функцией. Именно в нём производятся процессы ввода числа, сравнения его с заданным, обработки и генерации кол-ва жизней.

Функция **print_Results** выводит на экран консоли результат игры (игрок победил, проиграл или досрочно закончил).

Функция **more_Or_Less** служит функцией-подсказкой, которая выводит на экран консоли: введённое число оказалось больше заданного или меньше.

Функция **number_Of_Health** возвращает значение начального количества жизней.

Функция **print_Rules** выводит на экран консоли количество начальных жизней, условия выхода из игры и цель самой игры [4,5].

Программная реализация разработанного алгоритма.

Реализация данного алгоритма выполнена в виде программ, написанных на языке C++ среди Visual Studio 2019.

Тестирование программ проведено путем сравнения результатов её работы с контрольными примерами (рис. 1 - 3). Контрольные примеры разработаны с условием проверки полноты программы – рассмотрены все возможные исходы. Полученное решение соответствует всем рассмотренным в тестовых примерах случаям. Можно сделать вывод, что применяемые нами алгоритмы работоспособны, корректны а пример разработки тестовых вариантов может помочь другим студентам при решении подобных задач, по крайней мере, позволит убедить их в необходимости выполнения этого этапа работы.

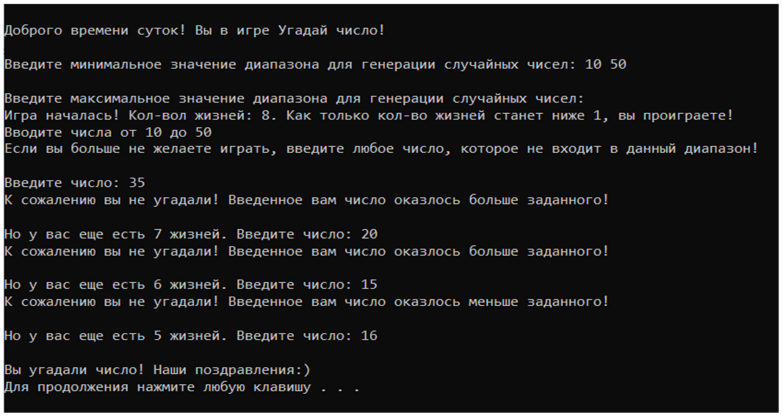


Рис. 1. Игрок угадывает число.

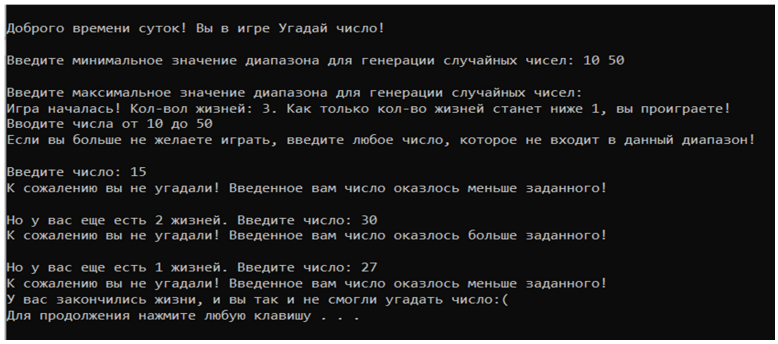


Рис. 2. Игра прерывается: израсходованы все жизни.

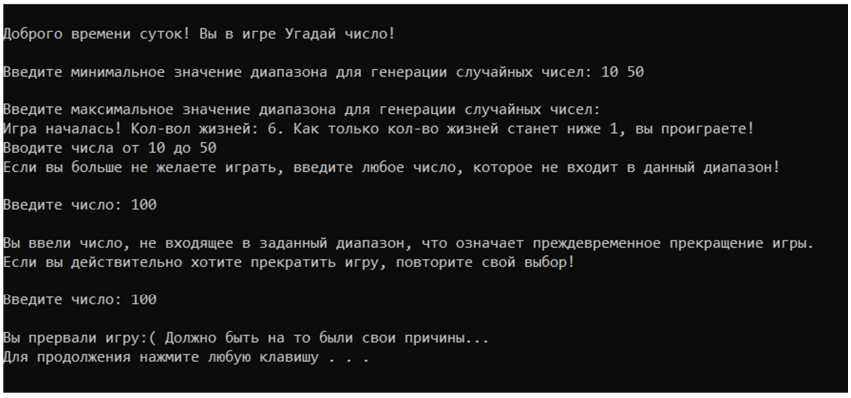


Рис. 3. Игра прерывается по желанию игрока.

Пользовательский интерфейс программы отображает необходимые шаги, которые пользователь/оператор должен предпринять для того, чтобы корректно решить поставленную задачу [6].

Заключение.

В статье продемонстрирован процесс разработки контрольного примера для задачи программирования компьютерной игры «Угадай число». Показаны основные моменты, на которые необходимо обратить внимание при разработке такого примера: словесно воспроизвести процесс игры, сопроводить его числовыми вычислениями, проверить все возможные исходы процесса. Показано, как работает программа, которая воспроизводит расписанный ранее алгоритм игры.

Литература:

1. Петрова Н.К., Конспект лекций «Программирование на С++». [Электронный ресурс]. URL: https://lms.kgeu.ru/course/view.php?id=2594
2. Ишмуратов Р.А., Ситников С.Ю. Применение визуальных сред разработки приложений для создания обучающих программ // Ученые записки ИСГЗ, Казань. - 2018. - Т.16. - №2, С.111-117.
3. Низамов А.А., Петрова Н.К. Сравнительный анализ языков Python и С++ для обучения студентов программированию. Modern Science. 2020. № 1-1. С. 322-325.
4. Губаев Т.О., Петрова Н.К. Разработка синтаксического анализатора арифметических выражений на языке С++ // Вестник Казанского государственного энергетического университета. 2018. Т. 10. № 2 (38). С. 32-40.
5. Куценко С.М. Электронные образовательные ресурсы как инструмент обучения / С.М. Куценко, В.В. Косулин // Вестник Казанского государственного энергетического университета. 2017. №4(36). С. 127-134.
6. Торкунова Ю.В., Шайдуллина Н.К. Электронная информационно-образовательная среда вуза: требования и возможности реализации // Ученые записки ИСГЗ. 2017. Т. 15. № 1. С. 541-546.