УДК 681.3.069

Автоматизация тестирования студенческих программ

© Ю.Е. Алексеев, А.В. Куров

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Рассмотрены вопросы автоматизации тестирования программ студентов в ходе выполнения лабораторных работ по курсу «Информатика». Предложенный подход позволяет студенту быстро протестировать программу заранее подготовленным и встроенным в его программу кодом, что обеспечивает более высокую интенсивность проводимых занятий, экономит время студента и преподавателя. Предлагаемая реализация ориентирована на проверку программ, написанных в среде VS C++.

***Ключевые слова*:** *тестирование программ, шаблон программы, файл шаблона, файл тестирования, технология тестирования.*

В настоящее время в условиях перехода к новым программам обучения студентов, предусматривающим сокращение количества аудиторных занятий, возникает необходимость интенсификации труда как преподавателей, так и студентов. Для решения этой задачи могут использоваться современные информационные технологии. Наиболее естественным это выглядит при изучении такой дисциплины, как информатика.

Выполняя лабораторные работы в компьютерных классах при изучении основ программирования по курсу «Информатика», студенты нередко представляют к защите программы, допускающие ошибки в вычислениях, несмотря на то, что на первых же занятиях (семинарах и лабораторных) они получают информацию по отладке программ и выполняют соответствующие работы. Поэтому преподавателю прежде всего приходится проверять соответствие разработанной программы заданию. В ряде случаев преподаватель может это сделать, рассматривая исходный текст программы, в ряде случаев — предложить студенту проверить работу программы на каком-либо наборе исходных данных. В любом случае это требует затрат времени и студента, и преподавателя.

В данной статье предлагается технология, позволяющая с минимальными затратами времени применить один из способов выявления ошибок в разрабатываемой студенческой программе — тестирование [1, 2]. Здесь тестирование будем понимать в узком смысле, а именно как проверку, выполняет ли программа свою функцию на некоторых заранее подготовленных или генерируемых при ее выполнении наборах данных. Известно, что в общем случае такое тестирование не гарантирует отсутствие ошибок в программе. Более того, в

учебных заданиях (что должно найти отражение и в программах) явно или по умолчанию при изучении определенной темы выдвигаются требования к использованию вполне определенных средств программирования и методов решения задач.

Суть предлагаемой технологии тестирования студенческих программ состоит в том, чтобы организовать работу программы в двух режимах: в обычном, без тестирования, и с тестированием (в этой статье будет рассмотрено тестирование только частей программы, обрабатывающих данные, но не ввод/вывод). Для достижения этой цели на структуру программы накладываются следующие основные ограничения:

– части программы должны быть функционально законченными;

– иметь только одну точку входа и одну точку выхода;

– вычислительные части программы не должны каким-либо образом менять входные данные;

– должны иметь вычисленные значения в точке выхода;

– не должны каким-либо образом, кроме как путем преобразования, менять переменные, являющиеся и входными и выходными.

Перечисленные ограничения отвечают принципам структурного программирования [3], и их применение облегчает, наряду с тестированием, отладку программ с использованием других методов.

Кроме того, имена всех переменных, используемых на входе и выходе в тестируемых частях программы, должны по написанию с учетом регистра, по типу и назначению совпадать с указанными в задании. Более того, допускается их описание, в том числе с начальными значениями, в шаблонах (о них речь пойдет ниже), подготавливаемом для разработки программ по учебным заданиям.

При создании нового проекта в консольном режиме VS C++ основная программа — файл с именем проекта и расширением .cpp изначально содержит стандартный шаблон, а разработка программы состоит в добавлении в стандартный шаблон операторов, объявлений, директив и т. д.

В предлагаемой технологии для автоматизации тестирования требуется заменить стандартный шаблон новым, специально разработанным для каждого задания. В новом шаблоне в минимальной реализации должны присутствовать:

– директива #define,

– одна или несколько директив #include,

– директивы условной компиляции,

– комментарии, сопровождающие эти директивы или имеющие иное назначение.

Для нового шаблона, как и для стандартного, запрещается редактировать, удалять и изменять положение директив и комментариев и других частей шаблона. Исключением является директива #define,

которая изначально закомментирована, т. е. представлена в виде однострочного комментария, что соответствует обычному, без тестирования, режиму работы программы.

Директивы #include содержат ссылки на файлы с добавляемыми в главную программу кодами тестирования. Эти директивы делят разрабатываемый код программы на части, реализующие отдельные подзадачи, которые явно формулируются в задании, если требуется соблюсти определенные требования (например, использовать определенный метод вычисления или определенным образом организовать ввод, вывод или задание значений данным), или по умолчанию, когда нет специальных требований (например, если в задании формулируется только, какие вычисления должна выполнять программа, то ввод исходных данных должен выполняться с клавиатуры и вывод результатов — в окно программы).

В новом шаблоне могут присутствовать отдельные комментарии, не связанные с директивами, обозначающие место начала и/или конца части кода программы. Таким образом, новый шаблон может дополнять задание.

Ниже представлен пример нового шаблона для задания, требующего последовательного ввода исходных данных, полной их обработки и вывода результатов.

```
#include "stdafx.h"
#include "conio.h"
#include "time.h"//для подпрограммы time в test3-3-1.txt
#include <windows.h> //для русских букв
//-#define test
int _tmain(int argc, _TCHAR* argv[])
{
//для использования при вводе/выводе русских букв сразу
//запустите программу с этим шаблоном без отладки
//и установите в окне программы шрифт Lucida Console
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
   double x=0.2, y;
#ifndef test
//Здесь разместите оператор ввода переменной x

#endif
#include "test3-3-1.txt"
//Здесь разместите операторы if else без использования
//логических функций для вычисления y=f(x)
#include "test3-3-1.txt"
//Здесь разместите операторы вывода данных,
//если ранее этого сделано не было

   getch();
   return 0;
}
```

Для примера взята первая часть задания 3.3.1 из [4]: требуется составить программу вычисления с помощью минимального числа операторов if else значения функции $Y(X)$ по формуле

$$Y(X)=\begin{cases}1 & \text{при } X<-2,\\ X/2 & \text{при } -2\le X\le 0,\\ \text{не определена} & \text{при } X=0,\\ 2 \text{ — в остальных случаях.}\end{cases}$$

К этому условию добавим, что тестирование следует проводить только после разработки части программы, вычисляющей $Y(X)$.

После копирования нового шаблона в файл главной программы с расширением .cpp в ее текст следует добавить:

– в первую часть программы оператор ввода значения переменной x:

```
scanf("%lf",&x);
```

– во вторую часть — операторы вычисления $Y(X)$:

```
if(x==0)
   printf("x==0, ФУНКЦИЯ НЕ ОПРЕДЕЛЕНА\n");
else
{
   if(x<-2)
   y=1;
   else
      if(x<0)
   y=x/2;
      else
         y=2;
}
```

– в третью часть программы оператор вывода значения переменной y:

```
   printf("x=%lf\ty=%lf\n",x,y);
```

Для этого примера использовался один файл тестирования вычислительной части программы. Ниже приведен его текст:

```
#ifdef test
 #ifndef test1
 #define test1
   double xx[5]={-3, -2, -1, 0, 1e-8}
    ,yy[5]={ 1, -1,-0.5, 0, 2}
```

```
     ,r;
    srand((unsigned)time( NULL ));
    r = rand();
    if(r==0)
       r=10;
    else if(r==RAND_MAX)
       r=r-10;
    xx[0]=-2.0-r;
  xx[2]=(-2.0+2.0/RAND_MAX*r)-1e-32;
  yy[2]=xx[2]/2.0;
  xx[4]=1e-32+r;
    for(int i=0; i<5;i++)
    {
       x=xx[i];
  #else
     if(x!=0)
          if(x==xx[i] && yy[i]!=y)
          {
             printf("ОШИБКА ПРИ x = %lf \n",x);
             printf("НАЖМИТЕ Enter\n");
             getch();
             return 0;
          }

    }//for
    printf("\nОШИБОК НЕ ОБНАРУЖЕНО!\n");
    printf("НАЖМИТЕ Enter\n");
#undef test1
    getch();
    return 0;
  #endif
#endif
```

Данные для тестирования заданы в массивах *xx* и *yy*, представляющих значения аргумента и функции соответственно, причем часть значений в массиве *xx* при выполнении программы заменяется случайными числами, принадлежащими диапазонам, в которых функция не меняет значения.

В случае обнаружения ошибки в окно программы будет выведен текст «ОШИБКА ПРИ x РАВНОМ» и значение *x*, при котором ошибка обнаружена, а работа программы будет закончена после нажатия клавиши Enter. Например, если это случится при *x* равном –2, то будет выведено сообщение:

```
ОШИБКА ПРИ x = -2.000000
```

Файл шаблона и файл тестирования следует при выдаче задания поместить в папку проекта, в которой расположен файл основной программы с расширением .cpp. Это упростит выдачу задания, а для

ссылки на файл тестирования в директивах #include достаточно будет указать только его имя.

В общем случае файлов тестирования может быть несколько, а заложенные в них коды и алгоритмы препроцессорной обработки могут определять как произвольные, так и вполне определенные последовательности разработки частей программы. Эти вопросы требуют отдельного рассмотрения.

ЛИТЕРАТУРА

[1] Канер С., Фолк Дж., Нгуен Е.К. *Тестирование программного обеспечения*. Киев, «ДиаСофт», 2001, 544 с.

[2] Винниченко И.В. *Автоматизация процессов тестирования*. Санкт-Петербург, Питер, 2005, 203 с.

[3] Дал У., Дейкстра Э., Хоор К. *Структурное программирование*. Москва, Мир, 1975, 247 с.

[4] Алексеев Ю.Е., Ваулин А.С., Куров А.В. *Практикум по программированию. Обработка числовых данных*. Трусов Б.Г., ред. Москва, Изд-во МГТУ им. Н.Э. Баумана, 2008, 288 с.

Статья поступила в редакцию 10.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Алексеев Ю.Е., Куров А.В. Автоматизация тестирования студенческих программ. *Инженерный журнал: наука и инновации*, 2013, вып. 6. URL: http://engjournal.ru/catalog/it/hidden/768.html

Алексеев Юрий Евтихович родился в 1938 г., окончил МВТУ им. Н.Э. Баумана в 1962 г. Канд. техн. наук, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Автор более 15 научных трудов в области программного обеспечения ЭВМ. e-mail: lekarru@mail.ru

Куров Андрей Владимирович родился в 1955 г., окончил. МВТУ им. Н.Э. Баумана в 1978 г. Канд. техн. наук, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии» МГТУ им. Н.Э. Баумана. Автор более 50 научных трудов в области надежности вычислительных систем и программного обеспечения ЭВМ. e-mail: avkur7@mail.ru