ЛАБОРАТОРНАЯ РАБОТА №4
Методы ручного контроля программного обеспечения

Ручной контроль обычно используют на ранних этапах разработки, т.к. возможность практической проверки подобных решений в этот период отсутствует. Путем безмашинного тестирования обнаруживают от 30 до 80% ошибок в профессиональных программах и до 100% - в учебных. К методам ручного контроля относятся: инспекции исходного текста, сквозные просмотры, проверка за столом, сухая прокрутка, символическая прокрутка, анализ ошибкоопасных мест, объяснение коллеге и некоторые другие.

Инспекции исходного текста представляют собой набор процедур и приемов обнаружения ошибок при изучении текста группой специалистов. В эту группу входят: автор программы, проектировщик, специалист по тестированию и координатор - компетентный программист, но не автор программы. Общая процедура инспекции предполагает следующие операции:

- участникам группы заранее выдается листинг программы и спецификация на нее;
- программист рассказывает о логике работы программы и отвечает на вопросы инспекторов;
- программа анализируется по списку вопросов для выявления исторически сложившихся общих ошибок программирования.

Кроме непосредственного обнаружения ошибок, результаты инспекции позволяют программисту увидеть другие сделанные им ошибки, получить возможность оценить свой стиль программирования, выбор алгоритмов и методов тестирования. Инспекция является способом раннего выявления частей программы, с большей вероятностью содержащих ошибки, что позволяет при тестировании уделить внимание именно этим частям.

Список вопросов для инспекций исходного текста зависит, как от используемого языка программирования, так и от специфики разрабатываемого программного обеспечения. Ниже приведен пример списка вопросов, который можно использовать при анализе правильности программ, написанных на языке Pascal,.

Список вопросов, для анализа правильности программ на языке Pascal

1. Контроль обращений к данным

- Все ли переменные инициализированы?
- Не превышены ли максимальные (или реальные) размеры массивов и строк?
- Не перепутаны ли строки со столбцами при работе с матрицами?
- Присутствуют ли переменные со сходными именами?
- Используются ли файлы? Если да, то при вводе из файла проверяется ли завершение файла?
- Соответствуют ли типы записываемых и читаемых значений?
- Использованы ли нетипизированные переменные, открытые массивы, динамическая память? Если да, то соответствуют ли типы переменных при «наложении» формата? Не выходят ли индексы за границы массивов?

2. Контроль вычислений

- Правильно ли записаны выражения (порядок следования операторов)?
- Корректно ли выполнены вычисления над неарифметическими переменными?
- Корректно ли выполнены вычисления с переменными различных типов (в том

числе с использованием целочисленной арифметики)?
- Возможно ли переполнение разрядной сетки или ситуация машинного нуля?
- Соответствуют ли вычисления заданным требованиям точности?
- Присутствуют ли сравнения переменных различных типов?

3. Контроль передачи управления

- Будут ли корректно завершены циклы?
- Будет ли завершена программа?
- Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? Корректно ли продолжатся вычисления?
- Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации «элемент найден» и «элемент не найден»?

4. Контроль межмодульных интерфейсов

- Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения?
- Не изменяет ли подпрограмма аргументов, которые не должны изменяться?
- Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами?

Сквозные просмотры осуществляются группой специалистов из 3-5 человек: координатор, секретарь (фиксирует ошибки), специалист по тестированию, программист (отвечает на вопросы о логике программы и принятых допущениях), независимый эксперт. Листинг программы и спецификация на нее анализируются на группе тестов. Участники заседания мысленно выполняют каждый тест в соответствии с логикой программы. При этом состояние программы (значения переменных) отслеживается на бумаге (доске).

Проверка за столом осуществляется тестировщиком (не автором программы), который проверяет текст программы по списку часто встречающихся ошибок и "пропускает" через программу тестовые данные.

Метод оценки программ направлен на повышение качества ПС и для поиска ошибок не применяется. Поэтому в данной лабораторной работе он не рассматривается.

Сухая прокрутка. Вручную моделируют работу ЭВМ при выполнении программ. Роль оперативной памяти играет трассировочная таблица. Ниже приведен пример **трассировочной таблицы** для программы, вычисляющей сумму квадратов n чисел.

n	sum	k
5		
	0	
		1
	1	
		2
	5	
		3
	14	

		4
	30	
		5
	55	

При **символической прокрутке** анализируется ход программы без подстановки конкретных значений переменных

Анализ ошибкоопасных мест. К ошибкоопасным ситуациям относятся:

1. *Обращение к данным:* инициализация и изменение переменных, схожесть их имен, индексация массивов, использование записей с вариантами, нетипизированных переменных, динамической памяти и т.п.

2. Вычисления: соответствие записи выражения и порядка его вычисления, логические выражения, операции сравнения, деления извлечения корня и др.;

3. Передача управления: развилки, циклы, вложенные условные операторы

4. Подпрограммы: количество, типы, порядок следования параметров, наличие ссылок;

5. Файлы: их создание, чтение, запись, завершение, типы.

Объяснение коллеге отличается от рассуждений про себя: принятые допущения и предположения должны прозвучать в явном виде.

Задачи

Рекомендации. При выполнении задач 4.1-4.3 результаты инспекции отражайте в таблице: ошибка –тип (строки программы можно пронумеровать). При выполнении задач 4.4-4.5 используйте трассировочную таблицу.

Задача 4.1. Выполнить инспекцию исходного текста программы на языке Паскаль.

```
Unit   NewParam;

interface

function ParamCount(): Word;
function ParamStr (Index: Integer): string;
function GetParamsStarting (Index: Integer): string;
implementation

uses   Objects,  TPString;

var
  CommandLine: PString;
  Params: array [Byte] of record L, R: Byte; end;
  LocParamCount: Byte;
```

```
function ParamCount;
begin
  ParamCount := LocParamCount;
end;

function ParamStr;
begin
  if (index = =0)
  then ParamStr := System.ParamStr (0)
  else with Params [Index-1] do ParamStr := Copy
(CommandLine^, L, R-L);
end;

function GetParamsStarting;
begin
  with Params [Index-1] do GetParamsStarting := Copy
(CommandLine^, L, $FF);
end;

const
  ParamDelims = [' ', #9];
  Quotes = ['"'];

var
  WaitForQuote: Boolean;
  B: Byte;
begin
  CommandLine := Ptr (PrefixSeg, $80);
  LocParamCount := 0;
  B := 0;
  while B <= Length (CommandLine^) do
  begin
    Inc (B);
    if CommandLine^ [B] in ParamDelims then Continue;
    with Params [LocParamCount] do
    begin
      WaitForQuote := CommandLine^ [B] in Quotes;
      if WaitForQuote then Inc (B)
      W:= B;
      while not (
        (B > Length (CommandLine^)) or
        WaitForQuote and (CommandLine^ [B] in Quotes)
or
        not WaitForQuote and (CommandLine^ [B] in
ParamDelims)) do
        Inc (B);
      R := B;
      if WaitForQuote then Inc (B);
```

```
      end;
      Inc (LocParamCount);
    end;
  end.
```

Задача 4.2. Выполнить инспекцию исходного текста программы.

```
program var1;
const x=12;
var
a : integer;
b: byte;
c: char;
arr: array [0..50] of  real;
f: file of char;
begin
   readln(a,b);
   if (c=x)
    begin
         a:=a+1;
         c:=c-“123”;
    end;
   for   a:=0 to 100 do
    begin
         arr[b]=sqrt(b);
         b:=b*3;
         a:=-b/2;
   end;
    assign(f, “test.txt”);
    {$I-};
    reset(f);
    {$I+};
    if (IOResult<>0) then writeln (файл f не
существует); else erase(f);
    if (a>0) then return 1
         else return –a;
end;
```

Задача 4.3. Выполнить инспекцию исходного текста программы.

```
 program var[2];
 const con=”222”;
 var
 a : integer;
 b: byte;
 c: char;
 b: boolean;
 arr: array [-10..10] of  real;
 procedure Srednee (var x1, x2, x3: integer)
 begin
   if (x1=x2=x3=0) then sred:=0
```

```
  else
  if (x2=x3=0) then sred:=x1;
       else
            if (x3=0) then c:=(x1+x2)/2
  else sred =(x1+x2+x3)/3;
return sred;
end;

begin
  b:=false;
  writeln ('Введите значения');
  readln(a,b);
  if (d>0)
  begin
       b:= Srednee (1,2,6);
  end;
  while not (b) do
  begin
  arr[d]:= arr[d]+4;
       d:=d+1;
  end;
  Srednee (d, arr[1], a);
  a:=sred;
end;
```

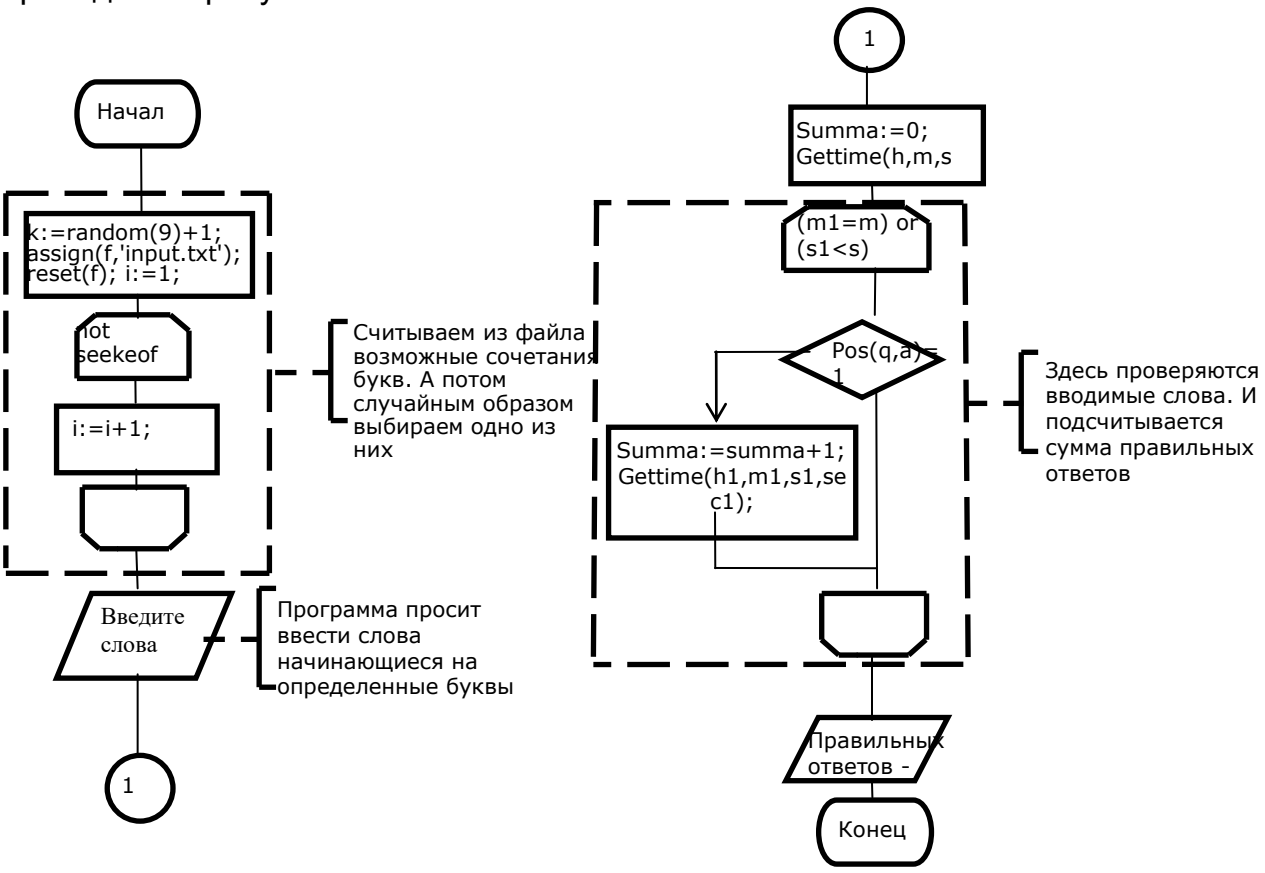
Задача 4.4. Составить тесты и осуществить сквозной просмотр программы, представляющей тест Ф.Лёзера «Узнавание чисел». На экране задерживаем на 40 секунд 20 случайно генерируемых чисел с их номерами по порядку. Затем пользователь должен ввести эти числа в том же порядке. Программа покажет, сколько чисел введено верно, и вычислит эффективность запоминания в процентах.

Текст программы:

```
Uses Crt;
Var a : array [1..20] of integer;
i,s,b : integer;
begin
randomize; clrscr;
for i:=1 to 20 do begin
a[i]:=random(100); writeln(i,'. ',a[i]);
end;
for i:=1 to 5 do delay(65000);
clrscr;
for i:=1 to 20 do begin
writeln(i,'. '); readln(b);
if b=a[i] then s:=s+1;
end;
clrscr;
writeln('Количество правильно названных чисел : ',s);
writeln('Эффективность запоминания
```

```
: ',(s*100)/20:2:0,' %');
readln;
end.
```

Задача 4.5. Составить тесты и осуществить сквозной просмотр программы для развития творчески Х способностей, алгоритм которой приведен на рисунке.



Текст программы:

```
program Creation;
uses dos;
var
f:text;
b:array[1..100] of string;
i,j,summa,k:integer;
h1,m1,sec1,s1,h,m,sec,s:word;
a,q:string;
begin
randomize;
k:=random(9)+1;
assign(f,'input.txt');
reset(f);
i:=1;
while not seekeof(f) do begin
readln(f,b[i]);
```

```
     i:=i+1;
     end;
     writeln(' Введите как можно больше существительных,
начинающихся на ',b[k]);
     q:=b[k];
     summa:=0;
     gettime(h,m,s,sec);
     while (m1=m) or (s1<s) do begin
     readln(a);
     if pos(q,a)=1 then summa:=summa+1;
     gettime(h1,m1,s1,sec1);
     end;
     writeln('Верных    слов    -    ',summa,'.    Мы    Вас
поздравляем!');
     readln;
     end.
```