БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ФНиДО

Специальность ПОИТ

Контрольная работа № 1

по дисциплине «Основы алгоритмизации и программирования»

часть 1

Вариант № 6

Выполнила студентка: Ващило А.Г.

Минск 2014

**1) Задача.**

Для целого числа М найти все простые множители в порядке их возрастания. Одинаковые множители выводить столько раз, сколько они встречаются. (Целое число Х является простым, если оно делится нацело только на единицу и само на себя).

**2). Схема алгоритма.**

На рис. 1, 2, 3 приведены три способа графического представления алгоритма.

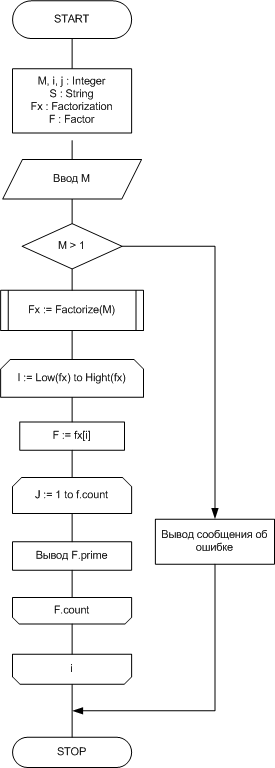


Рис. 1.1 Схема алгоритма по ГОСТ 19. 701-90

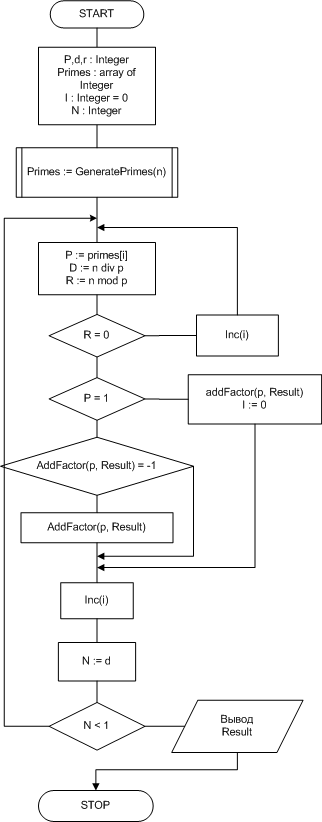


Рис. 1.2 Схема алгоритма функции **Factorize(n : Integer) : Factorization** по ГОСТ 19. 701-90

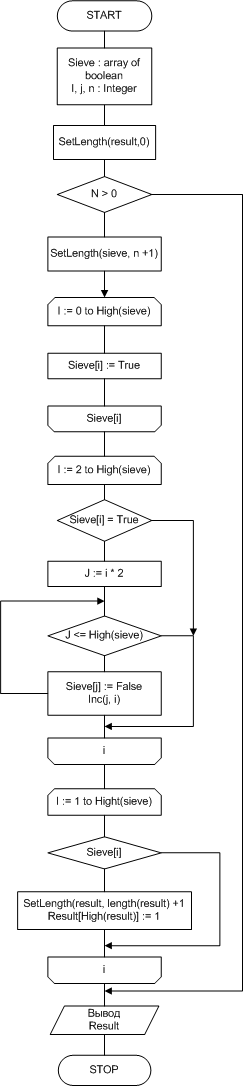


Рис. 1.3 Схема алгоритма функции **GeneratePrimes(n : Integer) : IntArray** по ГОСТ 19. 701-90

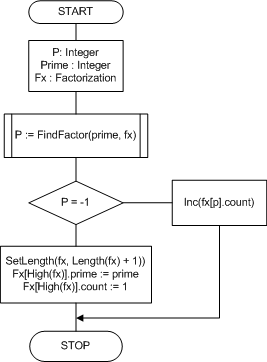


Рис. 1.4 Схема алгоритма процедуры

**AddFactor(prime: Integer; var fx: Factorization)** по ГОСТ 19. 701-90

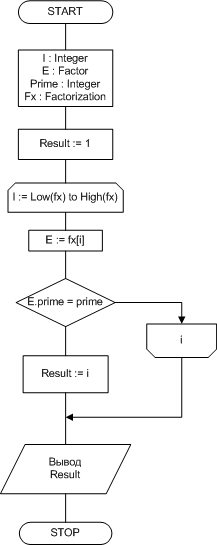


Рис. 1.5 Схема алгоритма функции

**FindFactor(prime: Integer; var fx: Factorization) : Integer** по ГОСТ 19. 701-90

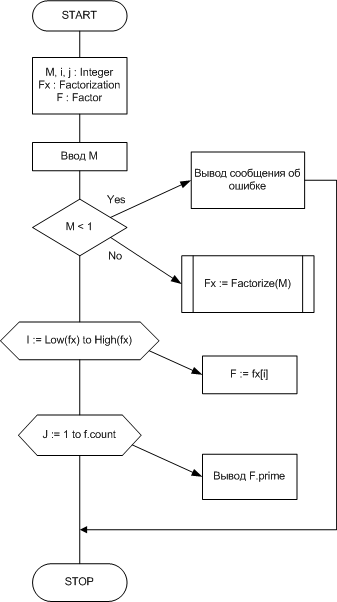


Рис. 2.1. Схема алгоритма, представленная методом Дамке

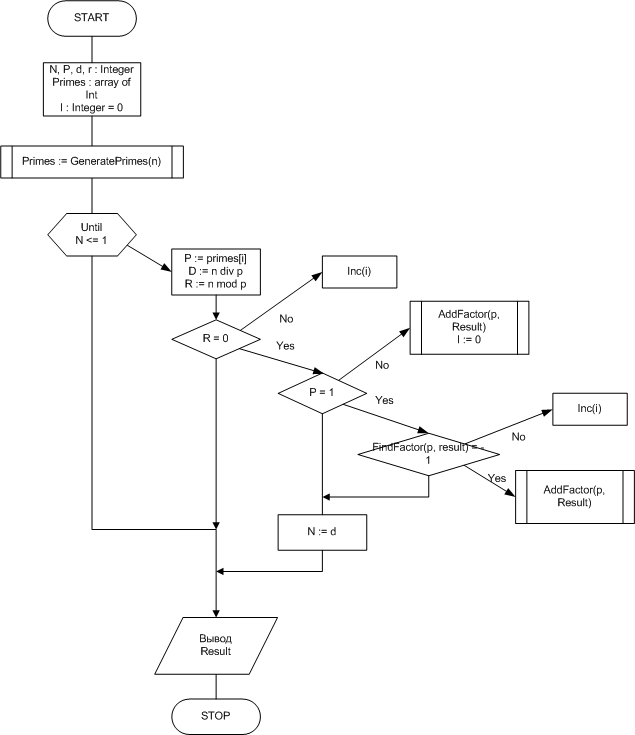


Рис. 2.2. Схема алгоритма функции Factorize, представленная методом Дамке

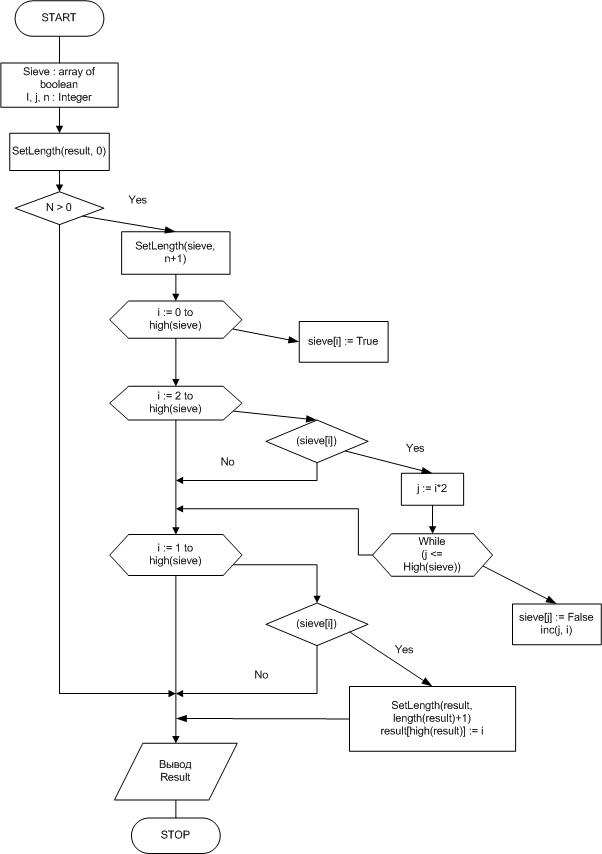


Рис. 2.3. Схема алгоритма функции GeneratePrimes, представленная методом Дамке

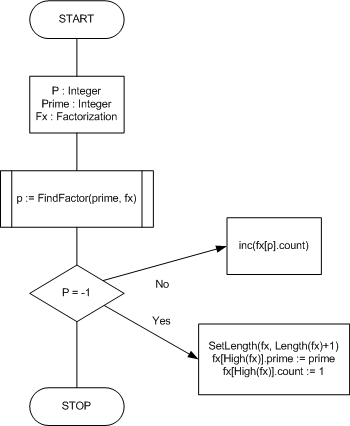


Рис. 2.4. Схема алгоритма процедуры AddFactor, представленная методом Дамке

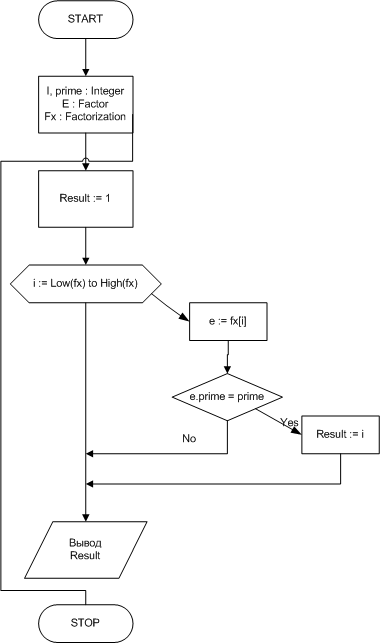


Рис. 2.5. Схема алгоритма процедуры FindFactor, представленная методом Дамке

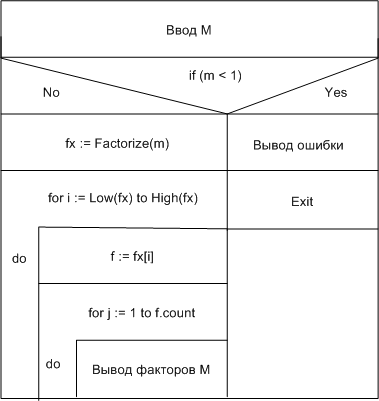


Рис. 3.1 Диаграмма Насси-Шнейдермана.

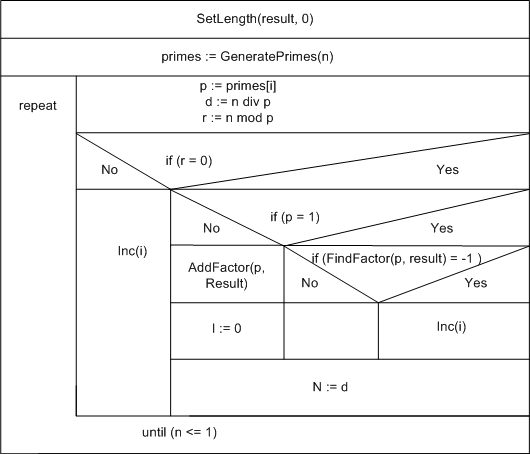


Рис. 3.2 Диаграмма функции Factorize, представленная методом Насси-Шнейдермана.

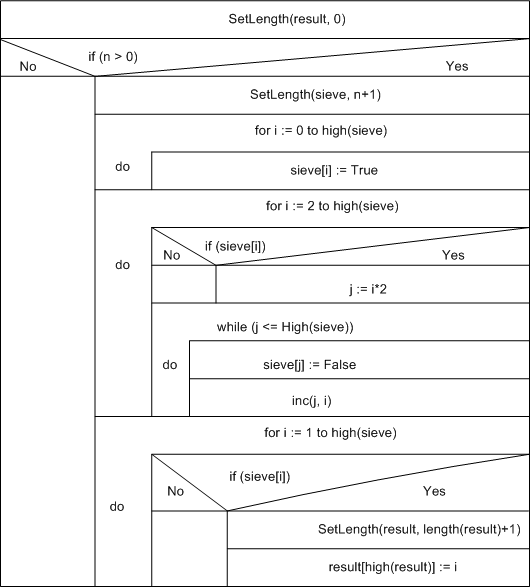


Рис. 3.3 Диаграмма функции GeneratePrimes, представленная методом Насси-Шнейдермана.

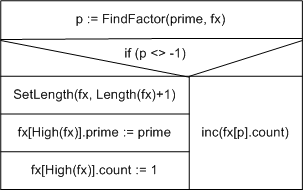


Рис. 3.4 Диаграмма процедуры AddFactor, представленная методом Насси-Шнейдермана.

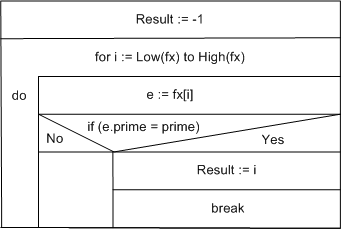


Рис. 3.5 Диаграмма функции FindFactor, представленная методом Насси-Шнейдермана.

**3). Листинг программы**

program kr1\_vashchilo;

uses

sysutils, strutils;

type

Factor = record

prime : Integer; {простой множитель}

count : Integer; {количество простых множителей}

end;

Factorization = array of Factor;

function FindFactor(prime: Integer; var fx: Factorization): Integer;

var

i: Integer;

e: Factor;

begin

Result := -1;

for i := Low(fx) to High(fx) do begin

e := fx[i];

if (e.prime = prime) then begin

Result := i;

break;

end;

end;

end;

procedure AddFactor(prime: Integer; var fx: Factorization);

var

p: Integer;

begin

p := FindFactor(prime, fx);

if (p <> -1) then begin

inc(fx[p].count);

end

else begin

SetLength(fx, Length(fx)+1);

fx[High(fx)].prime := prime;

fx[High(fx)].count := 1;

end;

end;

function GeneratePrimes(n : Integer) : IntArray;

var

sieve: array of boolean;

i,j : Integer;

begin

SetLength(result, 0);

if (n > 0) then begin

SetLength(sieve, n+1);

for i := 0 to high(sieve) do begin

sieve[i] := True;

end;

for i := 2 to high(sieve) do begin

if (sieve[i]) then begin

j := i\*2;

while (j <= High(sieve)) do begin

sieve[j] := False;

inc(j, i);

end;

end;

end;

for i := 1 to high(sieve) do begin

if (sieve[i]) then begin

SetLength(result, length(result)+1);

result[high(result)] := i;

end;

end;

end;

end;

function Factorize(n : Integer) : Factorization;

var

p, d, r : Integer;

i : Integer = 0;

primes : array of Integer;

begin

SetLength(result, 0);

primes := GeneratePrimes(n);

repeat

p := primes[i];

d := n div p;

r := n mod p;

if (r = 0) then begin

if (p = 1) then begin

if (FindFactor(p, result) = -1 ) then AddFactor(p, Result);

inc(i);

end

else begin

AddFactor(p, Result);

i := 0;

end;

n := d;

end

else begin

inc(i);

end;

until (n <= 1);

end;

var

m, i, j : Integer;

s : String;

fx: Factorization;

f: factor;

begin

write('Enter m: ');

readln(s);

m := StrToIntDef(s, 0);

if (m < 1) then begin

WriteLn('Error: m = ', m);

ReadLn;

Exit;

end;

fx := Factorize(m);

for i := Low(fx) to High(fx) do begin

f := fx[i];

for j := 1 to f.count do begin

Write(f.prime, ' ');

end;

end;

WriteLn('');

ReadLn;

end.

**4).Результат выполнения программы.**

