Опис методе класе

Стање сваког објекта класе дефинишу поља класе. Поља која одражавају стање које је јединствено називају се променљиве примерка. Када је декларисано, поље се може иницијализовати додељивањем вредности одговарајућег типа. Иницијализатор не мора бити константа, већ може бити друго поље, позив методе или неки израз који садржи ова два елемента, док год одговарају по типовима. Различите класе могу имати различит број и врсте поља. Вредности поља могу се дефинисати у фази израде апликације, али и касније, у фази извршавања, ако је алгоритамски предвиђено, односно, ако се та промена не би могла лоше одразити на ток извршења алгоритма. На пример, било би прилично непрофесионално дозволити кориснику да у току извршавања програма обрише едит за унос података или дугме за излазак из програма.

Класа, осим објеката, садржи и методе, односно, операције које су дефинисане са објектима те класе. Операције морају да се декларишу и дефинишу. Декларација мора садржати тип операције, назив и параметре са којима ради. Као метода може се дефинисати функција или процедура. Једним именом функције и процедуре зовемо потпрограмима зато што имају структуру било којег програма, али не могу да стоје самостално, јер су везане за објекат класе и извршавају се као реакција на неки догађај објекта. Потпрограми могу бити дефинисани и изван класе и не морају бити ни у каквој вези са објектима класе. Зато ћемо овде обрадити и функције и процедуре које нису методе класе. Структура је потпуно иста, разлика је само у власнику и параметрима функције или процедуре и у начину позивања. Методе класе позивају се као реакција на неки догађај, а позиви других функција или процедура пишу се у оквиру неке наредбе или су независне наредбе програма.

Врло често програм који пишемо постаје јако компликован, при чему се неки делови програма понављају са истим или сличним вредностима. Да би се поједноставило писање, читање и разумевање програма делови програма се издвајају у посебне релативно независне целине које се називају потпрограмима. Потпрограмима се придружује симболичко име помоћу којег ће се позивати из програма кад год је потребно извршити те издвојене радње. Потпрограми имају исту структуру као и сам програм, тј. имају заглавље и тело са наредбама. Потпрограми морају да се декларишу и дефинишу. Декларација се пише у заглављу програма, у делу за декларације, а дефиниције се пишу у делу за дефиниције (после службене речи ***implementation***). Разликујемо две врсте потпрограма: функције и процедуре.

Функције су једноставније. Декларишу се службеном речи ***function***, својим именом, аргументима са типом и типом функције, на пример:

```
function Stepen(b,e:real):real;
```

Декларација функције може да се напише у делу означеном са *private* или *public*. Приватне декларације видљиве су само из класе којој припадају, а јавне декларације су видљиве из читавог програма. Видљиве, значи да се могу позивати.

Функција може имати произвољан број аргумената који могу, а не морају бити истог типа. Они се називају још и улазним параметрима или улазним променљивама зато што вредност добијају у главном програму и ту своју вредност преносе у функцију. Осим улазних постоје и излазни параметри који своју вредност преносе из функције у главни програм. Функција може имати само један излазни параметар и то је само име функције (*зато функција мора имати декларисан тип јер се помоћу њеног имена нека вредност преноси у програм*). Да би се нека вредност из функције пренела у главни програм, имену функције се бар једном мора доделити вредност. У телу са наредбама име функције не сме да се нађе са десне стране знака додељивања нити у оквиру неке друге наредбе или израза, јер би то било позивање функције из саме функције, па би то била *рекурзивна функција*. Рекурзивне функције су специфичне функције које саме себе позивају. Коришћењем рекурзивних функција формално се избегава коришћење наредби циклуса, али како оне саме представљају једну нову врсту циклуса, то значи да њиховом употребом не избегавамо стварно коришћење циклуса, већ само користимо једну другачију врсту циклуса. При креирању рекурзивних функција мора се водити рачуна да

постоји најмање један коректан услов који ће одређивати колико пута ће она саму себе позивати. Ако се напише некоректан или погрешан услов програм неће давати исправне резултате или ће се *ублокирати*.

Функција мора да се дефинише да би могла да се користи у програму. Дефиниција функције је навођење радњи које се врше над неким подацима ради добијања неког резултата. Дефиниција функције се састоји из заглавља које мора бити идентично заглављу у декларацији функције и тела функције са наредбама, на пример:

```
function TForm1.Stepen(b,e:real):real;
var s:real;
begin s:=1;
      While e>0 do begin s:=s*b; e:=e-1;
                   end;
      stepen:=s;
end;
```

Једина разлика је ***TForm1.*** ознака која нам говори коме припада дефинисана функција.

Улазне променљиве у заглављу дефиниције морају бити истог имена и типа и написане истим редоследом као и у декларацији. Тип функције, такође мора бити идентичан типу функције у декларацији.

Осим улазних параметара, у функцији се могу користити и локалне и глобалне променљиве. Дефиниција функције се не сме преклапати са неком другом процедуром или функцијом, а име функције не сме бити исто као име неке друге функције, процедуре или променљиве. Име функције може се слободно изабрати при чему треба водити рачуна да се не користе резервисане речи и стандардни идентификатори. Обавезно је да име почне словом енглеске абецеде и не сме да садржи специјалне и знаке интерпункције.

Променљиве ***b*** и ***e*** у примеру декларације и дефиниције функције *Stepen* су формални параметри, тј. њихова функција је задовољење форме и немају никакву вредност. Када се функција позива из главног програма позива се са стварним, конкретним вредностима и зато се променљиве у позиву процедуре зову стварни параметри.

Функција се позива из главног програма навођењем имена и стварних параметара **УВЕК** из неке друге наредбе. Стварни параметри могу бити променљиве којима су додељене вредности у претходним наредбама програма, позиви неке друге дефинисане функције или конкретне вредности. *Једини услов за стварне параметре је да се **морају** слагати по типу са формалним параметрима*. Позив функције није независна наредба програма, а не може се у програму наћи ни са леве стране знака додељивања. Примери позива функције:

```
a:=Stepen(b,n);     <- одређује н-ти степен броја б
a:=Stepen(b,3);     <- одређује трећи степен броја б
a:=Stepen(2,4);     <- одређује четврти степен броја 2
```

Процедура је издвојени низ радњи који представља логичку целину посебно именовану ради позивања и извршавања по позиву у било којем делу програма. Процедуре се декларишу службеном речи ***procedure***, именом процедуре и улазним и излазним параметрима, на пример:

```
procedure Stepen(b,e:real;VAR s:real);
```

Процедура може имати произвољан број улазних и излазних параметара. Излазни се од улазних параметара одвајају службеном речи ***Var***. Улазни параметри вредност добијају у главном програму и ту вредност преносе у процедуру, вредност могу мењати у процедури, али се њихова нова вредност не преноси у главни програм. Излазни параметри процедуре своју вредност могу добијати у главном програму и уносити је у процедуру, могу добијати и мењати вредност унутар процедуре, а своју вредност преносе из процедуре у главни програм. Редослед параметара није предефинисан, тј. не морају сви улазни параметри бити на почетку, а излазни на крају листе. Могу се и наизменично писати један улазни, па један излазни, ако нам се тако свиђа, али је при томе, обавезно, да испред сваке листе излазних променљивих које су истог типа напишемо службену реч *Var*. Број улазних и излазних параметара није ограничен већ зависи од тога шта се обрађује и шта је резултат обраде у процедури. У телу са наредбама процедуре не сме се наћи име процедуре јер би то било позивање процедуре из саме процедуре, па би то онда била *рекурзивна процедура*. Рекурзивне процедуре су специфичне процедуре које саме себе позивају. Коришћењем рекурзивних процедура формално се избегава коришћење наредби циклуса, али како саме те процедуре представљају једну нову врсту циклуса, то значи да њиховом употребом не избегавамо стварно коришћење циклуса, већ само користимо једну другачију врсту циклуса. При креирању рекурзивних процедура мора се водити рачуна да постоји најмање један коректан услов који ће одређивати колико пута ће саму себе позивати. Ако се напише некоректан или погрешан услов програм неће давати исправне резултате или ће се *ублокирати*.

Процедура мора да се декларише и дефинише да би могла да се користи у програму. Декларација процедуре може да се напише у делу означеном са *private* или *public*. Приватне декларације видљиве су само из класе којој припадају, а јавне декларације су видљиве из читавог програма. Видљиве, значи да се могу позивати. Дефиниција процедуре је навођење радњи које се врше над неким подацима ради добијања неког резултата. Састоји се из заглавља које мора бити идентично декларацији процедуре и тела процедуре са наредбама. Улазне и излазне променљиве у заглављу дефиниције морају бити истог имена и типа и написане истим редоследом као и у декларацији. Пример дефиниције:

```
procedure TForm1.Stepen(b,e:real; VAR s:real);
begin s:=1;
      While e>0 do
            begin s:=s*b; e:=e-1;
            end;
end;
```

Једина разлика је ***TForm1.*** ознака која нам говори коме припада дефинисана функција.

Променљиве ***b*** и ***e*** су улазне, а и променљива ***s*** је излазни параметар процедуре. Осим улазних и излазних параметара, у процедури се могу користити и локалне и глобалне променљиве. Дефиниција процедуре се не сме преклапати са неком другом процедуром или функцијом, а име процедуре не сме бити исто као име неке друге функције, процедуре или променљиве. Име процедуре може се слободно изабрати при чему треба водити рачуна да се не користе резервисане речи и стандардни идентификатори. Обавезно је да име почне словом енглеске абецеде и не сме да садржи специјалне и знаке интерпункције.

Променљиве ***b***, ***e*** и ***s*** у заглављу декларације и дефиниције процедуре *Stepen* су формални параметри, тј. њихова функција је задовољење форме и немају никакву вредност. Када се процедура позива из главног програма позива се са стварним, конкретним вредностима и зато се променљиве у позиву процедуре зову стварни параметри.

Процедура се позива из главног програма навођењем имена и стварних параметара **УВЕК** као независна наредба програма. Стварни параметри могу бити променљиве којима су додељене вредности у претходним наредбама програма, претходно дефинисане функције или конкретне вредности. *Једини услов за стварне параметре је да се **морају** слагати по типу са формалним параметрима*. Позив процедуре је независна наредба програма и не може се наћи у оквиру неке друге наредбе или израза у програму. Улазни параметри у позиву процедуре могу бити променљиве којима су додељене вредности у претходним наредбама програма, позиви претходно дефинисаних функција или конкретне вредности, а излазни параметри увек **морају** бити променљиве. Примери позива процедуре:

```
Stepen(b,n,s);      <- одређује н-ти степен броја б
Stepen(b,3,s);      <- одређује трећи степен броја б
Stepen(2,4,s);      <- одређује четврти степен броја 2
```

У функцијама и/или процедурама могу се позивати друге функције и процедуре и користити њихове вредности да би се добили неки други резултати. Сваки задатак који може да се реши помоћу једне функције може да се реши и помоћу једне процедуре. Ако је задатак решен помоћу процедуре онда се он може решити и са онолико функција колико има излазних резултата у процедури (*али то понекад није једноставно јер излазни параметри могу бити зависни једни од других*). Зато се каже да су функције специјална врста процедуре које описују рачунање са једним излазним резултатом, а које се могу користити непосредно у формулама за израчунавање других вредности.

Тело функције или процедуре почиње службеном речи ***begin***, а завршава службеном речи ***end;*** између којих се налази потребан број наредби. Сва правила везана за тело програма важе и у овом случају, само што се, на крају програма ставља симбол *тачка*, а на крају функције или процедуре симбол *тачка-зарез*.

Променљиве које се користе у оквиру неке функције или процедуре могу бити глобалне, ако су декларисане у главном програму или локалне ако су декларисане у заглављу функције или процедуре. Глобалне променљиве своју вредност могу добијати и мењати било где у програму и преносе је из функције или процедуре у било који део програма. Ово, понекад, није практично јер се може десити да нам вредност променљиве добијена негде у програму треба и након повратка из функције или процедуре. Зато се чешће користе локалне променљиве. Оне своју вредност добијају у функцији или процедури у којој су декларисане и могу се користити само ту. Ако имамо више функција или процедура и у свима су декларисане променљиве истог имена њихове вредности су независне, не мешају се. Изван функције или процедуре немају никакву вредност и не могу се користити. Ако је у некој функцији или процедури декларисана локална

променљива чије име се поклапа са именом неке глобалне променљиве онда се у тој функцији или процедури не може користити вредност глобалне променљиве већ само локална и вредност локалне променљиве се ни на који начин не одражава на вредност глобалне.

Примери једноставнијих функција

- ***Факторијел унетог броја***

Улазни параметар функције је број - граница за који се рачуна факторијел. Због ограничења скупа целих бројева не можемо израчунати факторијел за сваки цео број. То ограничење треба уградити у главни програм, а функција треба да рачуна факторијел само за бројеве за које не прелази преко границе за целе бројеве (за бројеве веће од 20 функција не даје тачан резултат, да бисмо решили проблем са ограничењем целих бројева, можемо резултат прогласити реалним, али ни ово нам не помаже да добијемо тачан резултат код већих бројева јер факторијел неограничено расте, а меморија рачунара је ограничена).

```
function TForm1.Fakt(n:Integer):Int64;  // или :Real;
var i:Integer;f:Int64;  // или f:Real;
begin f:=1;
      For i:=1 to n do
          f:=f*i;
      fakt:=f;
end;
```

- ***Степеновање унетог броја другим унетим бројем***

Улазни параметри функције су број који се степенује (*основа*) и број којим се степенује (*експонент*). Због ограничења меморије не може се израчунати степен за сваки број на било који експонент. Разликујемо два случаја, када је експонент цео број и када је експонент реалан број. Прво решење је очигледније, друго решење подразумева коришћење логаритама и експоненцијалне функције јер се у програмирању не може израчунати корен било којег реда. Друго решење је општије, али користи експоненцијалну функцију и логаритме који се одређују са извесном грешком, може се десити да се добије резултат који је приближно тачан.

```
function TForm1.Step(a:Real; n:Integer):Real; // или Step(a,b:Real):Real;
var i:Integer;s:Real;
begin s:=1;
      For i:=1 to n do
          s:=s*a;       // или, уместо комплетног циклуса само: step:=exp(b*ln(a));
      step:=s;
end;
```

- ***Утврђивање да ли је уписани број прост***

Улазни параметар је број који тестирамо, а резултат је логичка вредност *true* ако је број прост или *false* ако број није прост:

```
function TForm1.Prost(n:Integer):boolean;
var d:integer;
begin If n<4
       then prost:=true              // сви мањи од 4 су прости бројеви
       else If not Odd(n)
             then prost:=false       // парни бројеви већи од 2 нису прости
             else begin d:=3;        // тражимо делиоце броја n
{ Пошто су бројеви непарни не могу бити дељиви парним бројем, зато је d:=3  најмањи потенцијални делилац.
  У циклусу проверавамо да ли је d делилац броја и да ли је мањи или једнак корену броја (јер је корен броја највећи
  делилац који не треба множити већим од њега да би се добио број). Број је прост ако нисмо нашли неки његов
  делилац док није d>Sqrt(n) }
                  While (n mod d<>0)and(d<=Sqrt(n)) do
                         d:=d+2;
                  prost:=d>Sqrt(n);
                end;
end;
```

Примери једноставнијих процедура

- ***Факторијел унетог броја***

Улазни параметар процедуре је број - граница за који се рачуна факторијел. Због ограничења скупа целих бројева не можемо израчунати факторијел за сваки цео број. То ограничење треба уградити у главни програм, а процедура треба да рачуна факторијел само за бројеве за које не прелази преко границе за целе бројеве (за бројеве веће од 20 функција не даје тачан резултат, да бисмо решили проблем са ограничењем целих бројева, можемо резултат прогласити реалним, али ни ово нам не помаже да добијемо тачан резултат код већих бројева јер факторијел неограничено расте, а меморија рачунара је ограничена).

```
procedure TForm1.Fakt(n:Integer; Var f:Int64); // или Var f:Real;
var i:Integer;
begin f:=1;
      For i:=1 to n do
          f:=f*i;
end;
```

- ***Степеновање унетог броја другим унетим бројем***

Улазни параметри процедуре су број који се степенује (*основа*) и број којим се степенује (*експонент*), а излазни параметар је *степен*. Због ограничења меморије не може се израчунати степен за сваки број на било који експонент. Разликујемо два случаја, када је експонент цео број и када је експонент реалан број. Прво решење је очигледније, друго решење подразумева коришћење логаритама и експоненцијалне функције јер се у програмирању не може израчунати корен било којег реда. Друго решење је општије, али користи експоненцијалну функцију и логаритме који се одређују са извесном грешком, може се десити да се добије резултат који је приближно тачан.

```
procedure TForm1.Step(a:Real; n:Integer; var s:Real); // или Step(a,n:Real; var s:Real);
var i:Integer;
begin s:=1;
      For i:=1 to n do
          s:=s*a;       // или, уместо комплетног циклуса само: step:=exp(b*ln(a));
end;
```

- ***Утврђивање да ли је уписани број прост***

Улазни параметар је број који тестирамо, а резултат је логичка вредност *true* ако је број прост или *false* ако број није прост:

```
procedure TForm1.Prost(n:Integer; Var p:boolean);
var d:integer;
begin If n<4
       then p:=true
       else If not Odd(n)
             then p:=false
             else begin d:=3;
                        While (n mod d<>0)and(d<=Sqrt(n)) do
                              d:=d+2;
                        p:=d>Sqrt(n);
                  end;
end;
```

Примери рекурзивних функција и процедура

Карактеристично за рекурзивну функцију или процедуру је постојање услова који одређује докле ће ће се рекузија понављати.

- ***Факторијел унетог броја***

```
function TForm1.Fakt(n:Integer):Int64;  // или :Real;
begin If n=0
       then fakt:=1
       else fakt:=fakt(n-1)*n;
end;
```

```
procedure TForm1.Fakt(n:Integer; Var f:Int64); //или Var f:Real);
begin If n=0
       then f:=1
       else begin fakt(n-1,f);f:=f*n;
            end;
end;
```

- ***Степеновање унетог броја другим унетим бројем***

```
function TForm1.Step(o,e:Integer):Integer;
begin If e=0
       then step:=1
       else step:=step(o,e-1)*o;
end;
```

```
procedure TForm1.Step(o,e:Integer; Var s:Integer);
begin If e=0
       then s:=1
       else begin step(o,e-1,s);s:=s*o;
            end;
end;
```

- ***Утврђивање да ли је уписани број прост***

Почетна вредност промењљиве ***d***, која представља потенцијалне делиоце броја који се тестира, у позиву функције или процедуре мора бити 2.

```
// рекурзивна функција позива саму себе док не нађе делиоца броја или док је d<=Sqrt(b)
function TForm1.Prost(b,d:Integer):Boolean;
begin prost:=true;
      If (b mod d<>0)and(d<=Sqrt(b))
       then prost:=Prost(b,d+1)
       else If b mod d=0
             then prost:=false;
end;
// рекурзивна процедура позива саму себе док не нађе делиоца броја или док је d<=Sqrt(b)
procedure TForm1.Prost(b,d:Integer; Var p:Boolean);
begin p:=true;
      If (b mod d<>0)and(d<=Sqrt(b))
       then Prost(b,d+1,p)
       else If b mod d=0
             then p:=false;
end;
```

Питања на која треба обратити пажњу

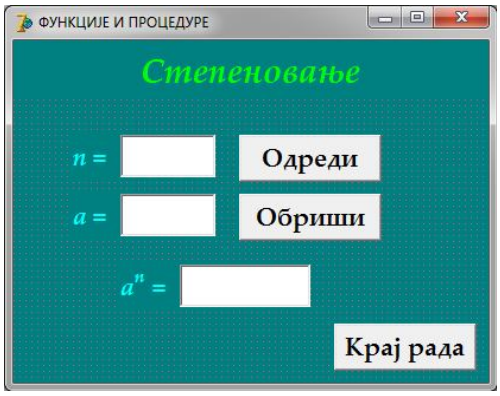
- Шта су методе класе?
- Декларација и дефиниција методе.
- Шта су функције?
- Врсте функција.
- Декларација функције.
- Дефиниција функције.
- Шта су процедуре?
- Врсте процедура.
- Декларација процедуре.
- Дефиниција процедуре.
- Разлика између функција и процедура.
- Разлика између *private* и *public* декларације.
- Параметри код функција.
- Параметри код процедура.
- Разлика између улазних и излазних параметара.
- Позивање функције.
- Позивање процедуре.
- Формални параметри.
- Стварни параметри.
- Локалне промењљиве.
- Глобалне промењљиве.

Задаци са функцијама и процедурама

Задаци који следе биће урађени на два начина: први ће бити сирово решавање задатка уз минимално коришћење уметничког надахнућа, а други начин ће бити покушај да неке од читалаца ових редова убеди да свет програмирања може бити цветни аранжман који креирамо срцем и свом својом душом. Задаци са функцијама и процедурама се решавају као и сви други, разлика је само што неке, погодне делове програма замењујемо позивима функција или процедура и на тај начин олакшавамо писање, читање и разумевање програма.
Први задатак ћемо објаснити детаљно да би се схватила суштина и предност коришћења кориснички дефинисаних функција и процедура, док ћемо наредне задатке решавати у *тишини*, односно, са много мање коментарисања, препуштајући читаоцима да сами дођу до неких, корисних закључака.

- ***Саставити програм који одређује n-ти степен уписног броја.***

Након креирања форме написаћемо комплетан програм:




```
// двокликом на први едит декларише се процедура
// брише резултат ако се промени експонент
// ова процедура није неопходна јер се контрола уноса и
// исписа остварује помоћу ReadOnly контрола едита
procedure TForm1.Edit1Change(Sender: TObject);
begin Edit3.Clear;
end;
// у Object Inspector-у на листићу Events дефинишемо
// реакцију на догађај OnKeyPress
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['-','0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Edit2.SetFocus;
end;
// у Object Inspector-у на листићу Events дефинишемо реакцију на догађај OnChange тако што
// из падајућег менија изаберемо већ дефинисану процедуру Edit1KeyPress на тај начин смо
// дефинисали да се резултат брише и ако се основа промени
// у Object Inspector-у на листићу Events дефинишемо реакцију на догађај OnKeyPress
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
// двокликом на тастер Крај рада декларише се процедура
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
// двокликом на тастер Обриши декларише се процедура
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;
      Edit1.SetFocus;
end;
// двокликом на тастер Одреди декларише се процедура
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
    s:real;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      s:=Stepen(b,Abs(n));
      If n<0
       then Edit3.Text:=Format('%1.5f',[1/s])
       else Edit3.Text:=Format('%1.0f',[s]);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
```

Функцију ***Stepen(b,n)*** ћемо прво декларисати у делу за декларације (испод службене речи ***private***). Ово морамо да урадимо сами јер функција није реакција на неки догађај, па преводилац не може да је препозна и декларише сам.

```
      private
         function Stepen(o,e:integer):integer;
```

Оваква декларација значи: функција има формалне целобројне параметре ***o*** - основа и ***e*** - експонент и целобројна је. Сада функцију треба дефинисати. То можемо урадити било где у телу програма, а најједноставније је пре самог краја програма (пре краја са тачком, испред ***end.*** додаћемо ред и уписати дефиницију функције). Комплетну дефиницију, и заглавље и тело функције са наредбама, такође, пишемо сами.

```
// функција за одређивање степена природног броја
function TForm1.Stepen(o,e:integer):integer;
var s:integer;
begin s:=1;
      While e>0 do
      begin s:=s*o;e:=e-1;
      end;
      stepen:=s;
end;
// рекурзивна функција за одређивање степена природног броја
function TForm1.Stepen(o,e:integer):integer;
begin If e=0
       then stepen:=1
       else stepen:=stepen(o,e-1)*o;
end;
```

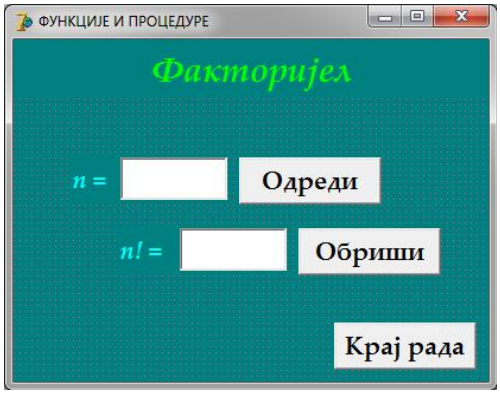
Приметили смо колико је рекурзивна функција једноставније записана, то је њена главна предност. Услов ***e=0*** је јако важан и одлучује да ли ће се функцији доделити вредност или ће се поново позвати. Понекад су рекурзивне функције много једноставније (*за писање, не увек и за разумевање*) и тада се оправдано користе, други пут су компликованије и нема оправдања за њихово коришћење. На програмеру је да се определи да ли ће и када користити рекурзивну функцију у решавању неког задатка.
Решићемо сада задатак са процедуром за степеновање.
Разлика је у декларацији и дефиницији процедуре за степеновање као и у позиву из главног програма, остатак програма је истоветан, па га нећемо поново исписивати.

```
// декларација процедуре Stepen
      private
         procedure Stepen(o,e:integer;var s:integer);
// дефиниција процедуре Stepen
procedure TForm1.Stepen(o,e:integer;var s:integer);
begin s:=1;
      While e>0 do
      begin s:=s*o;e:=e-1;
      end;
end;
// дефиниција рекурзивне процедуре Stepen
procedure TForm1.Stepen(o,e:integer;var s:integer);
begin If e=0
       then s:=1;
       else begin stepen(o,e-1,s);s:=s*o;
            end;
end;
// позивање процедуре Stepen
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
    s:real;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      Stepen(b,Abs(n),s);
      If n<0
       then Edit3.Text:=Format('%1.5f',[1/s])
       else Edit3.Text:=Format('%1.0f',[s]);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
```

Приметили смо да рекурзивна процедура није много једноставније записана, али то не мора бити увек тако. Понекад су рекурзивне процедуре много једноставније и тада се оправдано користе, други пут су компликованије и нема оправдања за њихово коришћење. На програмеру је да се определи да ли ће и када користити рекурзивну процедуру у решавању неког задатка.

- ***Саставити програм који одређује факторијел уписаног броја.***

Након креирања форме написаћемо комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
            then Edit2.SetFocus;
end;
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
            then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
    s:real;
begin n:=StrToIntDef(Edit1.Text,1);
      If n<0
       then n:=0;
      Edit1.Text:=IntToStr(n);
      f:=Fakt(n);
      Edit2.Text:=IntToStr(f);
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
```

Функцију ***Fakt(n)*** ћемо прво декларисати у делу за декларације.

```
      private
         function Fakt(n:integer):integer;
```

Сада ћемо ову функцију дефинисати.

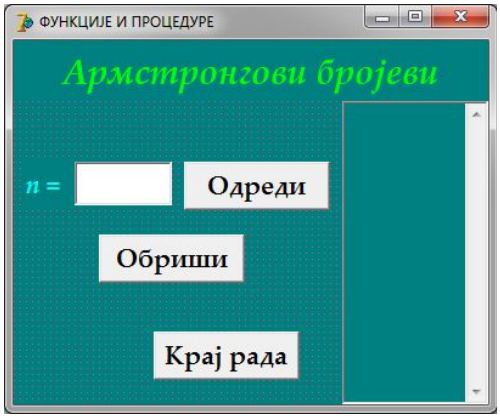
```
// функција за одређивање факторијела природног броја
function TForm1.Fakt(n:integer):integer;
var f:integer;
begin f:=1;
      While n>0 do
      begin f:=f*n;n:=n-1;
      end;
      fakt:=f;
end;
// рекурзивна функција за одређивање факторијела природног броја
function TForm1.Fakt(n:integer):integer;
begin If n=0
       then fakt:=1
       else fakt:=fakt(n-1)*n;
end;
```

Исто, са процедуром.

```
// декларација процедуре Fakt
      private
         procedure Fakt(n:integer;var f:integer);
// дефиниција процедуре Fakt
procedure TForm1.Fakt(n:integer;var f:integer);
begin f:=1;
      While n>0 do
      begin f:=f*n;n:=n-1;
      end;
end;
// дефиниција рекурзивне процедуре Fakt
procedure TForm1.Fakt(n:integer;var f:integer);
begin If n=0
       then f:=1
       else begin fakt(n-1);f:=f*n;
            end;
end;
// позивање процедуре Fakt
procedure TForm1.Button1Click(Sender: TObject);
var n,f:integer;
begin n:=StrToIntDef(Edit1.Text,1);
      If n<0
       then n:=0;
      Edit1.Text:=IntToStr(n);
      Fakt(n,f);
      Edit2.Text:=IntToStr(f);
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
```

- ***Саставити програм који одређује све Армстронгове бројеве до n.***

Армстронгови бројеви су они код којих је збир цифара степенованих бројем цифара једнак почетном броју. Јасно је да за једноцифрене нема смисла проверавати услов јер је $n^1=n$, па можемо рећи да сви једноцифрени бројеви испуњавају услов да буду Армстронгови. Да бисмо проверили услов за остале, вишецифрене бројеве декларисаћемо и дефинисати функцију за степеновање и функцију која рачуна збир степенованих цифара. У главном програму ћемо само позивати одговарајућу функцију и проверавати испуњеност услова. Армстронгове бројеве ћемо уписивати у мемо поље.
Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Edit2.SetFocus;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Memo1.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
begin n:=StrToIntDef(Edit1.Text,500);Edit1.Text:=IntToStr(n);
      For b:=1 to n do
       If (b<10)or(b=Cifre(b,Length(Edit1.Text)))
        then Memo1.Lines.Add(IntToStr(b));
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
function TForm1.Stepen(o,e:integer):integer;
var s:integer;
begin s:=1;
      While e>0 do
      begin s:=s*o;e:=e-1;
      end;
      stepen:=s;
end;
function TForm1.Cifre(n,m:integer):integer;
var c,z:integer;
begin z:=0;
      While n>0 do
      begin c:=n mod 10;
            n:=n div 10;
            z:=z+Stepen(c,m);
      end;
      cifre:=z;
end;
// Рекурзивна функција за одређивање збирова степена цифара
function TForm1.Cifre(n,m:integer):integer;
var c:integer;
begin If n=0
       then cifre:=0
       else begin c:=n mod 10;
                  cifre:=Cifre(n div 10)+Stepen(c,m);
            end;
end;
```

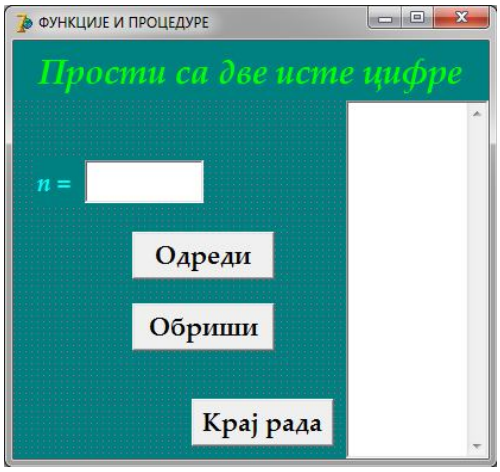
Исто, са процедуром:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,b,z:integer;
begin n:=StrToIntDef(Edit1.Text,500);Edit1.Text:=IntToStr(n);
      For b:=1 to n do
      begin If b>9
             then Cifre(b,Length(Edit1.Text),z)
             else z:=b;
            If b=z
             then Memo1.Lines.Add(IntToStr(b));
      end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
procedure TForm1.Cifre(n,m:integer;var z:integer);
var c:integer;
begin z:=0;
      While n>0 do
      begin c:=n mod 10;
            n:=n div 10;
            z:=z+Stepen(c,m);
      end;
end;
// Рекурзивна процедура за одређивање збира степена цифара
procedure TForm1.Cifre(n,m:integer;var z:integer);
var c:integer;
begin If n=0
       then z:=0
       else begin c:=n mod 10;
```

```
                Cifre(n div 10,z);
                z:=z+Stepen(c,m);
            end;
end;
```

Функцију *Stepen* смо већ описали као рекурзивну, а видели смо и како се степен може одредити помоћу процедуре и рекурзивне процедуре, па овде нису приказане.

- ***Саставити програм који исписује све просте бројеве мање од n са две исте цифре.***



Направићемо логичку функцију која одређује да ли је неки број прост и логичку функцију која проверава да ли број има две исте цифре. Јасно је да једноцифрени прости бројеви не могу имати две исте цифре, али и да, осим броја 11, нема двоцифрених простих са две исте цифре. Зато ћемо проверу почети од 101, наравно, ако је уписани број већи од 100.
Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Edit2.SetFocus;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Memo1.Clear;
      Edit1.SetFocus;
end;

procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      If n>11
       then Memo1.Lines.Add('11');
      b:=101;
      While b<=n do
      begin If (Prost(b)) and (Isto(b))
                then Memo1.Lines.Add(IntToStr(b));
            b:=b+1;
      end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;

// функција која проверава да ли је број прост
function TForm1.Prost(a:integer):boolean;
var d:integer;
begin d:=3;
      While (a mod d<>0)and(d<=Sqrt(a)) do
             d:=d+2;
      prost:=d>Sqrt(a);
end;

// функција која проверава да ли број има све различите цифре
function TForm1.Isto(a:integer):boolean;
var c:integer;
    sc:set of 0..9;
begin sc:=[a mod 10];
      a:=a div 10;
      isto:=false;
      Repeat c:=a mod 10;
             a:=a div 10;
             If c in sc
              then isto:=true
              else sc:=sc+[c];
      Until a=0;
end;

// рекурзивна функција која проверава да ли је број прост
function TForm1.Prost(b,d:integer):boolean;
begin prost:=true;
      If (b mod d<>0)and(d<=Sqrt(b))
       then prost:=Prost(b,d+2)
       else If b mod d=0
             then prost:=false;
end;
```

Ако решавамо задатак рекурзивном функцијом која одређује да ли је број прост позив функције треба изменити са ***Prost(b, 3)***, како бисмо задали почетну вредност за потенцијалне делиоце, односно:

```
        If (Prost(b,3)) and (Isto(b))
```

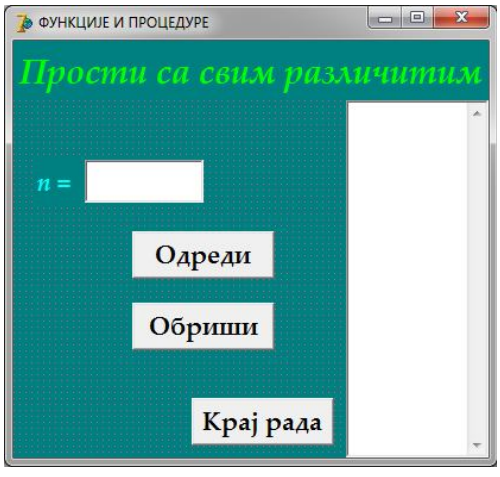
Исто, са процедурама:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
    p:boolean;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      If n>11
       then Memo1.Lines.Add('11');
      b:=101;
      While b<=n do
      begin Prost(b,p);
            If p
             then begin Isto(b,p);
                        If p
                         then Memo1.Lines.Add(IntToStr(b));
                  end;
            b:=b+2;
      end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
// процедура која проверава да ли је број прост
procedure TForm1.Prost(a:integer;var p:boolean);
var d:integer;
begin d:=3;
      While (a mod d<>0)and(d<=Sqrt(a)) do
             d:=d+2;
      p:=d>Sqrt(a);
end;
// рекурзивна процедура која проверава да ли је број прост
procedure TForm1.Prost(a,d:integer;var p:boolean);
begin If(a mod d<>0)and(d<=Sqrt(a))
       then Prost(a,d+2,p)
       else p:=d>Sqrt(a);
end;
// процедура која проверава да ли број има све различите цифре
procedure TForm1.Isto(a:integer;var p:boolean);
var c:integer;
    sc:set of 0..9;
begin sc:=[a mod 10];
      a:=a div 10;
      p:=true;
      Repeat c:=a mod 10;
             a:=a div 10;
             If c in sc
              then p:=false
              else sc:=sc+[c];
      Until a=0;
end;
```

Процедуру и функцију ***Isto***, такође, можемо написати као рекурзивне, али оне не постају ништа једноставније, па на томе овог пута нећемо инсистирати.

- ***Саставити програм који исписује све просте бројеве мање од n са свим различитим цифрама.***

Направићемо логичку функцију која одређује да ли је неки број прост и логичку функцију која проверава да ли број има истих цифара. Јасно је да једноцифрени прости бројеви не могу имати истих цифар, али и да, осим броја 11, сви двоцифрени прости имају две различите цифре. Зато ћемо проверу почети од 101, наравно, ако је уписани број већи од 100.



Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Edit2.SetFocus;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Memo1.Clear;
      Edit1.SetFocus;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      If n>11
       then Memo1.Lines.Add('11');
      b:=101;
      While b<=n do
      begin If (Prost(b)) and (Isto(b))
                then Memo1.Lines.Add(IntToStr(b));
            b:=b+2;
      end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
// функција која проверава да ли је број прост
function TForm1.Prost(a:integer):boolean;
var d:integer;
begin d:=3;
      While (a mod d<>0)and(d<=Sqrt(a)) do
             d:=d+2;
      prost:=d>Sqrt(a);
end;
// функција која проверава да ли број има све различите цифре
function TForm1.Isto(a:integer):boolean;
var c:integer;
    sc:set of 0..9;
begin sc:=[a mod 10];
      a:=a div 10;
      isto:=true;
      Repeat c:=a mod 10;
             a:=a div 10;
             If c in sc
              then isto:=false
              else sc:=sc+[c];
      Until a=0;
end;
// рекурзивна функција која проверава да ли је број прост
function TForm1.Prost(b,d:integer):boolean;
begin prost:=true;
      If (b mod d<>0)and(d<=Sqrt(b))
       then prost:=Prost(b,d+2)
       else If b mod d=0
             then prost:=false;
end;
```

Ако решавамо задатак рекурзивном функцијом која одређује да ли је број прост позив функције треба изменити са ***Prost(b, 3)***, како бисмо задали почетну вредност за потенцијалне делиоце, односно:

```
If (Prost(b,3)) and (Isto(b))
```

Исто, са процедурама:

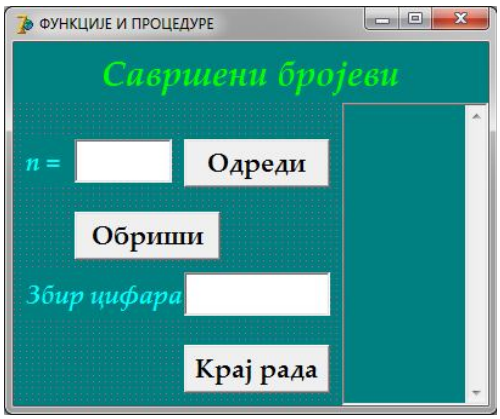
```
procedure TForm1.Button1Click(Sender: TObject);
var n,b:integer;
    p:boolean;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      If n>11
       then Memo1.Lines.Add('11');
      b:=101;
      While b<=n do
      begin Prost(b,p);
            If p
             then begin Isto(b,p);
                        If p
                         then Memo1.Lines.Add(IntToStr(b));
                  end;
            b:=b+2;
      end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
// процедура која проверава да ли је број прост
procedure TForm1.Prost(a:integer;var p:boolean);
var d:integer;
begin d:=3;
      While (a mod d<>0)and(d<=Sqrt(a)) do
             d:=d+2;
      p:=d>Sqrt(a);
end;
// рекурзивна процедура која проверава да ли је број прост
procedure TForm1.Prost(a,d:integer;var p:boolean);
begin If(a mod d<>0)and(d<=Sqrt(a))
       then Prost(a,d+2,p)
```

```
       else p:=d>Sqrt(a);
end;
// процедура која проверава да ли број има све различите цифре
procedure TForm1.Isto(a:integer;var p:boolean);
var c:integer;
    sc:set of 0..9;
begin sc:=[a mod 10];
      a:=a div 10;
      p:=true;
      Repeat c:=a mod 10;
             a:=a div 10;
             If c in sc
              then p:=false
              else sc:=sc+[c];
      Until a=0;
end;
```

Процедуру и функцију ***Isto***, такође, можемо написати као рекурзивне, али оне не постају ништа једноставније, па на томе ни овог пута нећемо инсистирати.

- ***Саставити програм који одређује збир цифара савршених бројева до n.***

Направићемо логичку функцију која одређује да ли је неки број савршен и целобројну функцију која одређује збир цифара броја. Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:



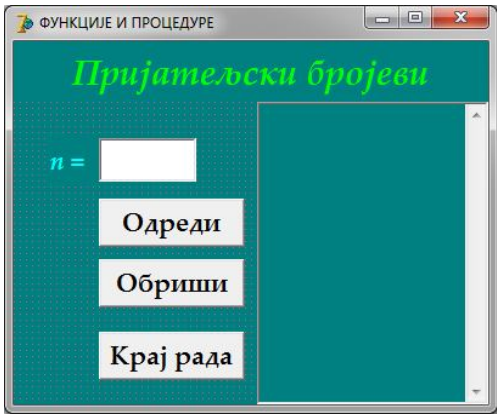

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;
      Memo1.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,i,z:integer;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      z:=0;
      For i:=1 to n do
       If Savrsen(i)
        then begin Memo1.Lines.Add(IntToStr(i));
                   z:=z+ZCifara(i);
      end;
      Edit2.Text:=IntToStr(z);
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
// функција која проверава да ли је број савршен
function TForm1.Savrsen(b:integer):boolean;
var s,d:integer;
begin s:=1;
      For d:=2 to b div 2 do
       If b mod d=0
        then s:=s+d;
      savrsen:=s=b;
end;
// функција која одређује збир цифара броја
function TForm1.ZCifara(b:integer):integer;
var s:integer;
begin s:=0;
      Repeat s:=s+b mod 10;
             b:=b div 10;
      until b=0;
      zcifara:=s;
end;
// рекурзивна функција која одређује збир цифара броја
function TForm1.ZCifara(b:integer):integer;
begin If b=0
       then zcifara:=0
       else zcifara:=b mod 10+zcifara(b div 10);
end;
```

Исто, са процедурама:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,i,z,s:integer;
    p:boolean;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      z:=0;
      For i:=1 to n do
      begin Savrsen(i,p);
            If p
             then begin Memo1.Lines.Add(IntToStr(i));
                        ZCifara(i,s);
                        z:=z+s;
                  end;
      end;
      Edit2.Text:=IntToStr(z);
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
      Button2.SetFocus;
end;
// процедура која проверава да ли је број савршен
procedure TForm1.Savrsen(b:integer; var p:boolean);
var d,s:integer;
begin s:=1;
      For d:=2 to b div 2 do
       If b mod d=0
        then s:=s+d;
      p:=s=b;
end;
// процедура која одређује збир цифара броја
procedure TForm1.ZCifara(b:integer; var s:integer);
begin s:=0;
      While b>0 do
      begin s:=s+b mod 10;
            b:=b div 10;
      end;
end;
```

- ***Саставити програм који одређује парове пријатељских бројева до n.***

Направићемо целобројну функцију која одређује збир делилаца броја и позиваћемо је за тестирани број и збир његових делилаца. Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Memo1.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,a,s,b:integer;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
      For a:=1 to n do
      begin b:=ZDelilaca(a);
            s:=ZDelilaca(b);
            If (s=a)and(a<b) then
             Memo1.Lines.Add(Format('%5d%5d',[a,b]));
      end;
end;
// функција која одређује збир делилаца броја
function TForm1.ZDelilaca(b:integer):integer;
var s,d:integer;
begin s:=1;
      For d:=2 to b div 2 do
       If b mod d=0
        then s:=s+d;
      zdelilaca:=s;
end;
```

Исто са процедуром:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,a,s,b:integer;
begin n:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(n);
```

```
      For a:=1 to n do
      begin ZDelilaca(a,b);
            ZDelilaca(b,s);
            If (s=a)and(a<b)
             then Memo1.Lines.Add(Format('%5d%5d',[a,b]));
      end;
end;
// процедура која одређује збир делилаца броја
procedure TForm1.ZDelilaca(b:integer;var s:integer);
var d:integer;
begin s:=1;
      For d:=2 to b div 2 do
       If b mod d=0 then s:=s+d;
end;
// рекурзивна процедура која одређује збир делилаца броја
procedure TForm1.ZDelilaca(b,d:integer;var s:integer);
begin If d<2 then s:=1
      else begin ZDelilaca(b,d-1,s);
                 If b mod d=0 then s:=s+d;
           end;
end;
```

Ако се користи рекурзивна процедура онда треба променити њене позиве у:

```
ZDelilaca(a,a div 2,b);
ZDelilaca(b,b div 2,s);
```

- ***Саставити програм који одређује НЗС и НЗД за два уписана броја.***

Направићемо неколико верзија целобројних функцију која ће одређивати НЗС и НЗД да бисмо што више прихватили начин решавања проблема функцијама. Наравно, довољно је имати само једну од приказаних функција.

Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Edit2.SetFocus;
end;
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;Edit4.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var a,b,s:integer;
begin a:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(a);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      s:=NZS(a,b);Edit3.Text:=IntToStr(s);
      s:=NZD(a,b);Edit4.Text:=IntToStr(s);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
// функција за одређивање НЗС са циклусом For
function TForm1.NZS(a,b:integer):integer;
var i:integer;
begin For i:=a*b downto a do
       If (i mod a=0)and(i mod b=0)
        then nzs:=i;
end;
// функција за одређивање НЗС са циклусом While
function TForm1.NZS(a,b:integer):integer;
var s:integer;
begin s:=a;
      While s mod b<>0 do s:=s+a;
      nzs:=s;
end;
```

```
// функција за одређивање НЗС са циклусом Repeat
function TForm1.NZS(a,b:integer):integer;
var s:integer;
begin s:=a*b+a;
      Repeat s:=s-a;
      until s-a mod b<>0;
      nzs:=s;
end;
// функција за одређивање НЗД са циклусом For
function TForm1.NZD(a,b:integer):integer;
var i:integer;
begin For i:=1 to a do
       If (a mod i=0)and(b mod i=0) then nzd:=i;
end;
// функција за одређивање НЗД са циклусом While
function TForm1.NZS(a,b:integer):integer; // Euklidov algoritam
var s:integer;
begin While (a mod b<>0)and(b mod a<>0) do
      begin If a>b
             then a:=a-b
             else b:=b-a;
      end;
      If a<b
       then nzs:=a
       else nzs:=b;
end;
// рекурзивна функција за одређивање НЗД са циклусом Repeat
function TForm1.NZD(a,b:integer):integer; // Euklidov algoritam
var s:integer;
begin s:=Abs(a-b);
      If a<b
       then nzd:=NZD(a,s)
       else If a>b
        then nzd:=NZD(s,b)
        else nzd:=a;
end;
```

Уместо две функције могли смо задатак решити и помоћу једне функције и особине НЗС и НЗД овако:

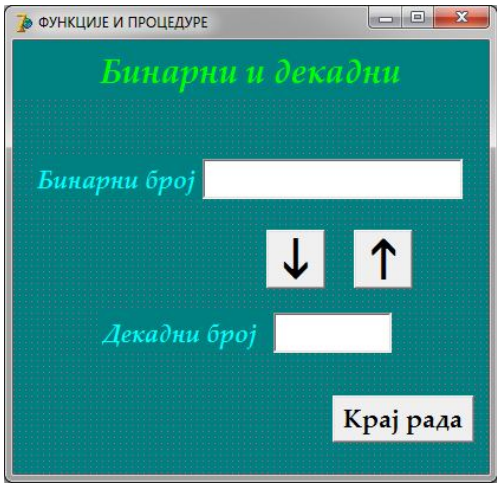
```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,d:integer;
begin a:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(a);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      d:=NZD(a,b);Edit4.Text:=IntToStr(d);
      Edit3.Text:=IntToStr(a*b div d);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
// или овако
procedure TForm1.Button1Click(Sender: TObject);
var a,b,d:integer;
begin a:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(a);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      d:=NZS(a,b);Edit3.Text:=IntToStr(d);
      Edit4.Text:=IntToStr(a*b div d);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
```

Исти, са процедуром:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,d:integer;
begin a:=StrToIntDef(Edit1.Text,1);Edit1.Text:=IntToStr(a);
      b:=StrToIntDef(Edit2.Text,1);Edit2.Text:=IntToStr(b);
      NZSiD(a,b,c,d);
      Edit3.Text:=IntToStr(c);Edit4.Text:=IntToStr(d);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
// један од начина записа процедуре за НЗС и НЗД
procedure TForm1.NZS(a,b:integer; VAR c,d:integer); // Euklidov algoritam
begin c:=a*b;
      While (a mod b<>0)and(b mod a<>0) do
      begin If a>b
             then a:=a-b
             else b:=b-a;
      end;
      If a<b
       then d:=a
       else d:=b;
      c:=c div d;
end;
```

- ***Саставити програм који декадни број претвара у бинарни или обрнуто.***

Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button2.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
```

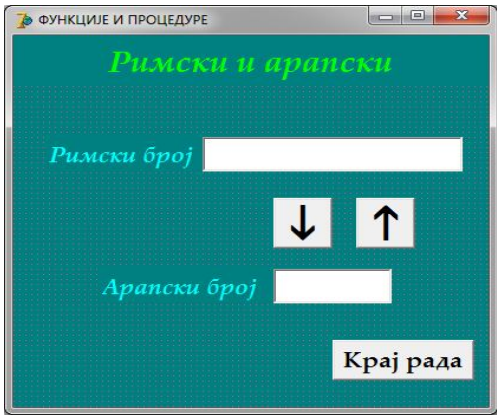
Клик мишем у првом или другом едиту брише садржај оба едита:

```
procedure TForm1.Edit1Enter(Sender: TObject);
begin Edit1.Clear;Edit2.Clear;
end;
// превођење бинарног у декадни број - стрелица на доле
procedure TForm1.Button1Click(Sender: TObject);
var a:string;
begin a:=Edit1.Text;
      Edit2.Text:=IntToStr(Dekadni(a));
end;
function TForm1.Dekadni(a:string):integer;
var s,i,d:integer;
begin s:=1;d:=0;
      For i:=Length(a) downto 1 do
      begin d:=d+StrToInt(a[i])*s;
            s:=s*2;
      end;
      dekadni:=d;
end;
// превођење декадног у бинарни број - стрелица на горе
procedure TForm1.Button2Click(Sender: TObject);
var d:integer;
begin d:=StrToIntDef(Edit2.Text,0);
      Edit1.Text:=Binarni(d);
end;
function TForm1.Binarni(d:integer):string;
var b:string;
begin b:='';
      While d>0 do
      begin b:=IntToStr(d mod 2)+b;
            d:=d div 2;
      end;
      binarni:=b;
end;
// превођење бинарног у декадни број рекурзивном процедуром:
procedure TForm1.Button1Click(Sender: TObject);
var a:string;
    d:integer;
begin a:=Edit1.Text;
      d:=0;
      dekadni(a,1,d);
      Edit2.Text:=IntToStr(d);
end;
procedure TForm1.Dekadni(a:string;s:integer;var d:integer);
begin if length(a)=1
       then d:=d+StrToInt(a)*s
       else begin d:=d+StrToInt(a[Length(a)])*s;
                  dekadni(Copy(a,1,Length(a)-1),s*2,d);
            end;
end;
// превођење декадног у бинарни број рекурзивном процедуром
procedure TForm1.Button2Click(Sender: TObject);
var a:string;
    b:integer;
begin b:=StrToIntDef(Edit2.Text,0);
      Edit2.Text:=IntToStr(b);
      Binarni(b,a);
      Edit1.Text:=a;
end;
```

```
procedure TForm1.Binarni(b:integer;var a:string);
begin If b=0
       then a:=''
       else begin binarni(b div 2,a);
                  a:=a+IntToStr(b mod 2);
            end;
end;
```

- ***Саставити програм који римски број претвара у арапски или обрнуто.***

Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['I','V','X','L','C','D','M',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button2.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
```

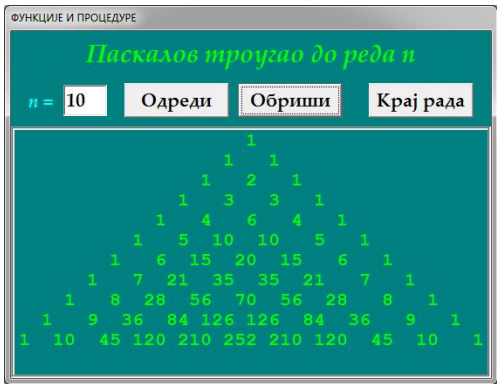
Клик мишем у првом или другом едиту брише садржај оба едита:

```
procedure TForm1.Edit1Enter(Sender: TObject);
begin Edit1.Clear;Edit2.Clear;
end;
// превођење римског у арапски број функцијом - стрелица на доле
procedure TForm1.Button1Click(Sender: TObject);
var a:string;
begin a:=Edit1.Text;
      If length(a)>0
       then begin a:=Arapski(a);
                  If a=' '
                   then ShowMessage('Greska u unosu broja')
                   else Edit2.Text:=a;
            end;
end;
function TForm1.Arapski(a:string):string;
var i,b:integer;
    g:boolean;
begin g:=false;
      b:=0;i:=0;
      Repeat i:=i+1;
             case a[i] of
              'M':If b mod 1000=0
                   then If b<3000
                         then b:=b+1000
                         else g:=true
                   else If b mod 1000=100
                         then b:=b+800
                         else g:=true;
              'C':If b mod 100=0
                   then If(b mod 1000<300)or(b mod 1000>400)and(b mod 1000<800)
                         then b:=b+100
                         else g:=true
                   else If b mod 100=10
                         then b:=b+80
                         else g:=true;
              'X':If b mod 10=0
                   then If(b mod 100<30)or(b mod 100>40)and(b mod 100<80)
                         then b:=b+10
                         else g:=true
                   else If b mod 10=1
                         then b:=b+8
                         else g:=true;
              'D':If b mod 1000=0
                   then b:=b+500
                   else If b mod 1000=100
                         then b:=b+300
                         else g:=true;
              'L':If b mod 100=0
                   then b:=b+50
                   else If b mod 100=10
                         then b:=b+30
                         else g:=true;
```

```
              'V':If b mod 10=0
                   then b:=b+5
                   else If b mod 10=1
                         then b:=b+3
                         else g:=true;
              'I':If(b mod 10<3)or(b mod 10>4)and(b mod 10<8)
                   then b:=b+1
                   else g:=true;
             else g:=true;
            end;
      Until i=length(a);
      if g
       then arapski:=' '
       else arapski:=IntToStr(b);
end;
// превођење арапског у римски број рекурзивном процедуром - стрелица на горе
procedure TForm1.Button2Click(Sender: TObject);
var a:string;
    b:integer;
begin b:=StrToIntDef(Edit2.Text,1);
      If (b>3999)or(b<1) then b:=3999;
      Edit2.Text:=IntToStr(b);
      Rimski(b,0,a);
      Edit1.Text:=a;
end;
procedure TForm1.Rimski(a,c:integer;var r:string);
var c1,c2,c3:string;
begin If a>0 then
      begin c:=c+1;
            case c of
             1 : begin c1:='I';c2:='V';c3:='X';
                 end;
             2 : begin c1:='X';c2:='L';c3:='C';
                 end;
             3 : begin c1:='C';c2:='D';c3:='M';
                 end;
             4 : begin c1:='M';c2:=' ';c3:=' ';
                 end;
            end;
            case a mod 10 of
             1 : r:=c1+r;
             2 : r:=c1+c1+r;
             3 : r:=c1+c1+c1+r;
             4 : r:=c1+c2+r;
             5 : r:=c2+r;
             6 : r:=c2+c1+r;
             7 : r:=c2+c1+c1+r;
             8 : r:=c2+c1+c1+c1+r;
             9 : r:=c1+c3+r;
            end;
            Rimski(a div 10,c,r);
      end;
end;
```

- ***Саставити програм који исписује Паскалов троугао до реда n.***

Функцијом *BK(n,k)* одредићемо к-ти биномни коефицијент у н-том реду, а н-ти ред ћемо формирати стринг функцијом *Red(n)* која ће н пута позивати претходну функцију. Редове ћемо уписивати у мемо поље. Најпре ћемо креирати форму као на слици, а затим написати комплетан програм:




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button2.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Memo1.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,p:integer;
begin p:=StrToIntDef(Edit1.Text,10);  // овако креирана форма може да прихвати само 10 редова
      If p>10 then p:=10;             // ограничење се може укинути ако се повећа ширина форме
      Edit1.Text:=IntToStr(p);
```

```
      For n:=0 to p do
       Memo1.Lines.Add(Red(n));
      Edit1.ReadOnly:=false;
      Button2.SetFocus;
end;
function TForm1.Red(n:integer):string;
var k:integer;
    a:string;
begin a:='1';
      For k:=1 to n do
        a:=a+Format('%4d',[BK(n,k)]);
      red:=a;
end;
function TForm1.BK(n,k:integer):integer;
var i:integer;
    b:real;
begin b:=1;
      For i:=1 to k do
        b:=b*(n-i+1)/i;
      bk:=Trunc(b);
end;
```

Функцију за одређивање биномног коефицијента можемо написати и као рекурзивну:

```
function TForm1.BK(n,k:integer):real;
begin If k=0
       then BK:=1
       else BK:=BK(n,k-1)*(n-k+1)/k;
end;
```

Задатак се може решити и са три функције, овако:

```
function TForm1.Red(n:integer):string;
var k:integer;
    a:string;
begin a:='';
      For k:=0 to n do
        a:=a+Format('%1.0f',[BK(n,k)])+' ';
      red:=a;
end;
function TForm1.BK(n,k:integer):real;
begin bk:=Fakt(n)/(Fakt(k)*Fakt(n-k));
end;
function TForm1.Fakt(n:integer):real;
var f:real;
    i:integer;
begin f:=1;
      For i:=1 to n do
        f:=f*i;
      fakt:=f;
end;
```

Функција за одређивање факторијела може бити и рекурзивна:

```
function TForm1.Fakt(n:integer):real;
begin If n=0
       then fakt:=1
       else fakt:=fakt(n-1)*n;
end;
```

- ***Саставити програм који са тастатуре прихвата n бројева, а затим одређује број са највећим и број са најмањим збиром цифара и њихове редне бројеве, као и аритметичку средину уписаних бројева.***

Променљиве ***k, s, n, max*** и ***min*** декларисаћемо као глобалне јер се користе у различитим процедурама, а треба да преносе чувају своју вредност у читавом програму. Прва променљива ће бити бројач унетих бројева. Друга променљива је збир унетих. Трећа је граница за унос бројева и уноси се у први едит и не сме бити нула. Почетна вредност четврте променљиве је -1 јер је то мање од најмањег могућег збира цифара неког броја, па када се било који број упише он ће бити са већим збиром цифара и задатак ће увек имати коректно решење (*замислимо да смо за почетну вредност узели 20, а може се десити да приликом уноса бројева не унесемо ниједан број са више од две цифре, тј. највећи могући збир цифара унетих бројева је 18, па се никада неће десити да унети број има већи збир цифара од 20 и ниједан број се неће исписати као решење задатка што није коректно*). Пета променљива има почетну вредност ***maxint*** (*највећи цео број*) јер било који уписани број има мањи збир цифара и задатак ће увек имати

коректно решење. Креираћемо форму као на слици, а затим написати комплетан програм. Неким глобалним промењљивама ћемо дати предефинисану вредност у заглављу програма:

```
var
  k:integer = 0; s:integer = 0;
  n:integer;
  max:integer = -1; min:integer = maxint;
  Form1: TForm1;
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button2.Click;
end;
```

Ове две процедуре можемо заменити једном, нешто мало компликованијом:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then If sender=edit1
             then Button1.Click
             else Button2.Click
       else If not (key in ['0'..'9',#8,#128])
        then key:=#27
end;
```

Пошто немамо тастер за брисање, његову фукцију ће преузети процедура - реакција на измену садржаја првог едита:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin Edit2.Clear;Edit3.Clear;Edit4.Clear;
      Edit5.Clear;Edit6.Clear;Edit7.Clear;
      Memo1.Clear;
      k:=0;s:=0;
      max:=-1;min:=maxint;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin n:=StrToIntDef(Edit1.Text,1);
      If n=0 then n:=1;
      Edit1.Text:=IntToStr(n);
      Edit1Change(sender);
      Edit1.Enabled:=false;Button1.Enabled:=false;
      Edit2.Enabled:=true; Button2.Enabled:=true;
      Edit2.SetFocus;
end;
procedure TForm1.Button2Click(Sender: TObject);
var b,z:integer;
begin b:=StrToIntDef(Edit2.Text,0);Edit2.Text:=IntToStr(b);
      Memo1.Lines.Add(Edit2.Text);
      k:=k+1;s:=s+b;
      z:=ZbirCifara(b);
      If max<z     // одређивање броја са највећим збиром цифара
       then begin max:=z;
                  Edit3.Text:=IntToStr(b);
                  Edit4.Text:=IntToStr(k);
            end;
      If min>z     // одређивање броја са најмањим збиром цифара
       then begin min:=z;
                  Edit5.Text:=IntToStr(b);
                  Edit6.Text:=IntToStr(k);
            end;
      If n=k       // крај уноса бројева јер су сви унешени
       then begin Edit1.Enabled:=true; Button1.Enabled:=true;
                  Edit2.Enabled:=false;Button2.Enabled:=false;
                  Edit7.Text:=FloatToStr(s/n);
                  Edit1.SetFocus;
            end
       else begin Edit2.Clear;Edit2.SetFocus;
            end;
end;
function TForm1.ZbirCifara(a:integer):integer;
var z:integer;
begin z:=0;
      Repeat z:=z+a mod 10;
```

```
            a:=a div 10;
      until a=0;
      zbircifara:=z;
end;
```

Ова функција може се написати и као рекурзивна:

```
function TForm1.ZbirCifara(a:integer):integer;
begin If a=0
       then zbircifara:=0
       else zbircifara:=ZbirCifara(a div 10)+a mod 10;
end;
```

Ако два уписана броја имају исти збир и то је најмањи или највећи збир цифара као резултат ће се исписати онај који је први унет. Заменом релације < релацијом <= и релације > релацијом >= исписаће се последњи унети број што је једнако оправдано. Коректније би било да се испишу сви бројеви који имају исти, највећи или најмањи збир цифара, али то би било јако компликовано извести са овим нивоом знања програмирања.

- ***Уписују се бројеви све док се не упише 0. Саставити програм који одређује број са највећим и број са најмањим збиром цифара и њихове редне бројеве, као и аритметичку средину позитивних уписаних бројева.***

Задатак врло сличан претходном. Основна и највећа разлика је што је овде граница за упис бројева непозната. Може се уписати један, а може се уписати и хиљаду бројева, то се унапред не зна. Променљиве ***s, n, k, max*** и ***min*** декларисаћемо као глобалне јер се користе у различитим процедурама, а треба да преносе чувају своју вредност у читавом програму. Прва променљива је збир унетих бројева и на почетку мора бити 0. Друга је бројач унетих бројева и на почетку је, такође, 0. Трећа променљива броји позитивне уписане бројеве и почетна вредност јој је, такође, 0. Почетна вредност четврте променљиве је ***-1***, а пете ***maxint*** тако ће задатак увек имати коректно решење. Креираћемо форму као на слици, а затим написати комплетан програм. Свим глобалним променљивама доделићемо предефинисану вредност у заглављу програма:

```
var
  n:integer = 0; k:integer = 0; max:integer = -1;
  s:integer = 0; min:integer = maxint;
  Form1: TForm1;
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['-','0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit3.Clear;Edit4.Clear;
      Edit5.Clear;Edit6.Clear;
      Button1.Enabled:=true;
      Button2.Enabled:=false;
      Memo1.Clear;
      n:=0; max:=-maxint;
      s:=0; min:=maxint;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var b,z:integer;
begin b:=StrToIntDef(Edit1.Text,0);Edit1.Text:=IntToStr(b);
      If b=0
       then begin Edit1.ReadOnly:=true;
                  Button1.Enabled:=false;
                  Button2.Enabled:=true;
                  Button2.SetFocus;
            end
       else begin Memo1.Lines.Add(Edit1.Text);
                  n:=n+1;z:=ZbirCifara(b);
                  If max<z    // одређивање броја са највећим збиром цифара
                   then begin max:=z;
                              Edit2.Text:=IntToStr(b);
                              Edit3.Text:=IntToStr(k);
                        end;
```

```
                If min>z    // одређивање броја са најмањим збиром цифара
                 then begin min:=z;
                            Edit4.Text:=IntToStr(b);
                            Edit5.Text:=IntToStr(k);
                      end;
                If b>0
                 then begin s:=s+b;k:=k+1;
                            Edit6.Text:=Format('%1.5f',[s/n]);
                      end;
                Edit1.Clear;Edit1.SetFocus;
           end;
end;
function TForm1.ZbirCifara(a:integer):integer;
var z:integer;
begin z:=0;
      Repeat z:=z+a mod 10;
             a:=a div 10;
      until a=0;
      zbircifara:=z;
end;
```

Ова функција може се написати и као рекурзивна:

```
function TForm1.ZbirCifara(a:integer):integer;
begin If a=0
       then zbircifara:=0
       else zbircifara:=ZbirCifara(a div 10)+a mod 10;
end;
```

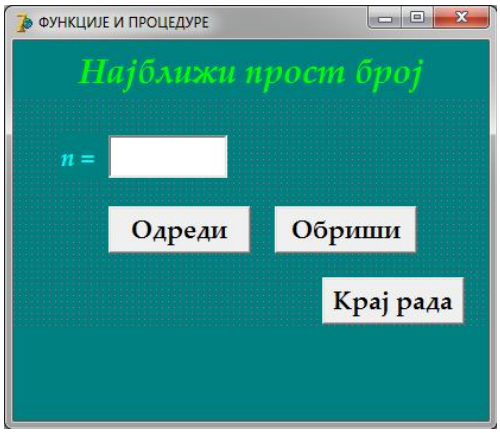
Ако два уписана броја имају исти збир и то је најмањи или највећи збир цифара као резултат ће се исписати онај који је први унет. Заменом релације < релацијом <= и релације > релацијом >= исписаће се последњи унети број што је једнако оправдано. Коректније би било да се испишу сви бројеви који имају исти, највећи или најмањи збир цифара, али то би било јако компликовано извести са овим нивоом знања програмирања.

Уместо иницијалног додељивања вредности глобалним променљивама могли смо дефинисати процедуру у којој се ради то исто, а која се покреће исцртавањем форме на екрану:

```
procedure TForm1.FormCreate(Sender: TObject);
begin n:=0; k:=0;  max:=-1;
      s:=0; min:=maxint;
end;
```

- ***Саставити програм који за унети број одређује најближи прости број.***

Ако је уписани број прост он је самом себи најближи. Ако број није прост проверићемо за један мањи и за један већи број. Ако је један од њих прост он је најближи. Ако су оба проста оба су најближа и треба их исписати оба. Ако ниједан није прост онда ћемо мањи смањити за један, а већи повећати за један и радити нову проверу за ова два броја. Поступак понављамо све док не нађемо бар један прост број који је онда најближи уписаном броју. Проверу, да ли је број прост обрадићемо логичком функцијом. Креираћемо форму као на слици, а затим написати комплетан програм.




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['0'..'9',#8,#13,#128])
       then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Label3.Caption:='';
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var i,b:integer;
    p,q:boolean;
begin b:=StrToIntDef(Edit1.Text,0);Edit1.Text:=IntToStr(b);
      p:=false;q:=p;
      i:=1;
      If Prost(b)
       then Label3.Caption:='Уписани број је најближи прост.'
       else While (not p) and (not q) do
            begin p:=Prost(b-i);
                  q:=Prost(b+i);
```

```
                If p
                 then If q
                  then Label3.Caption:='Најближи прости су '+Format('%1d %1d',[b-i,b+i])
                  else Label3.Caption:='Најближи прост је '+IntToStr(b-i)
                 else If q
                  then Label3.Caption:='Најближи прост је '+IntToStr(b+i)
                  else i:=i+1;
           end;
      Edit1.ReadOnly:=true;
      Button2.SetFocus;
end;
```

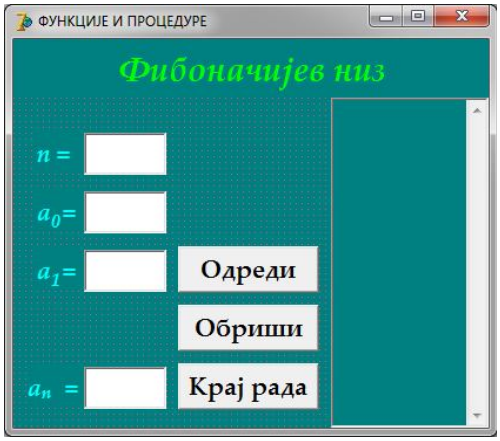
Логичка функција *Prost* тестира да ли је број прост:

```
function TForm1.Prost(n:integer):boolean;
var d:integer;
begin If n<4
       then prost:=true
       else If Odd(n)
        then begin d:=3;
                   While (n mod d<>0)and(d<=Sqrt(n)) do d:=d+2;
                   prost:=d>Sqrt(n);
             end
        else prost:=false;
end;
```

Задатак се може решити и са процедуром уместо функције, али се не добија ништа једноставније. Такође, задатак би се могао незнатно убрзати ако бисмо водили рачуна да ли је број паран или непаран, па да се променљива мења за 2 уместо за 1. Но, све то не би битно поправило програм, али би га укомпликовало.

- ***Саставити програм који исписује n-ти елемент низа чија су прва два елемента дати бројеви, а сваки следећи елемент се одређује као збир претходна два (Фибоначијев низ).***

Ако би почетне вредности два броја биле 0 и 1 онда би то био Фибоначијев низ у оригиналном опису. Овако само много личи. Испис елемената низа у мемо поље није неопходан, али је згодно видети шта програм ради. Елементе низа одредићемо уз помоћ процедуре. Креираћемо форму као на слици, а затим написати комплетан програм.

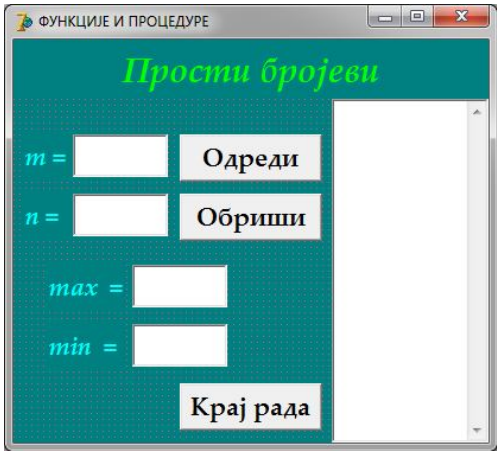



```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then If sender=edit1
             then Edit2.SetFocus
             else If sender=edit2
                   then Edit3.SetFocus
                   else Button1.SetFocus
       else If not (key in ['0'..'9',#8,#128])
             then key:=#27
end;

procedure TForm1.Edit2Enter(Sender: TObject);
var n:integer;
begin n:=StrToInt(Edit1.Text);
      If n=0
       then Edit1.SetFocus;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;Edit3.ReadOnly:=false;
      Edit4.Clear;Memo1.Clear;
      Edit1.SetFocus;
end;

procedure TForm1.Button1Click(Sender: TObject);
var a,b,n:integer;
begin n:=StrToIntDef(Edit1.Text,2);Edit1.Text:=IntToStr(n);
      a:=StrToIntDef(Edit2.Text,0);Edit2.Text:=IntToStr(a);
      b:=StrToIntDef(Edit3.Text,1);Edit3.Text:=IntToStr(b);
      Memo1.Lines.Add(IntToStr(a));
      Memo1.Lines.Add(IntToStr(b));
      Label8.Caption:=IntToStr(n);
      Odredi(n,a,b);
      Edit4.Text:=IntToStr(b);
      Edit1.ReadOnly:=true;
      Edit2.ReadOnly:=true;
      Edit3.ReadOnly:=true;
      Button2.SetFocus;
end;
```

```
procedure TForm1.Odredi(n,a:integer; VAR b:integer);
var i,p:integer;
begin For i:=2 to n do
      begin p:=a+b;a:=b;b:=p;
            Memo1.Lines.Add(IntToStr(p));
      end;
end;
procedure TForm1.Odredi(n,a:integer; VAR b:integer); // рекурзивна процедура
var p:integer;
begin p:=a;
      If n>1
       then begin b:=a+b;
                  Memo1.Lines.Add(IntToStr(b));
                  Odredi(n-1,b-p,b);
            end;
end;
// уместо процедуре, може се искористити и ова, прилично једноставна рекурзивна функција
function TForm1.Odredi(n,a,b:integer):integer;
begin If n>1
       then begin Memo1.Lines.Add(IntToStr(a+b));
                  odredi:=Odredi(n-1,b,a+b);
            end
       else odredi:=b;
end;
```

- ***Саставити програм који исписује просте бројеве од m до n код којих је цифра прва са леве стране већа од 6, а прва са десне стране мања од 4, а затим исписати бројеве са највећим и најмањим збиром цифара.***

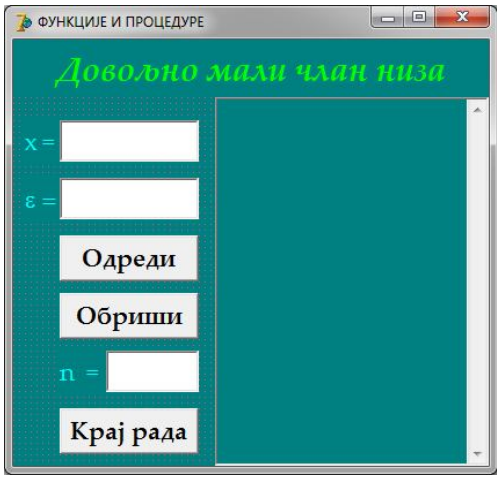
Задатак ћемо решити коришћењем две функције: логичке функције *Prost* која одређује да ли је неки број прост и целобројне функције *ZbirC* која одређује збир цифара броја. Измена садржаја било ког од прва два едита брише садржаје друга два едита и мемо поља. Креираћемо форму као на слици, а затим написати комплетан програм.




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then If sender=edit1
             then Edit2.SetFocus
             else Button1.Click
       else If not (key in ['0'..'9',#8,#128])
             then key:=#27
end;
procedure TForm1.Edit1Change(Sender: TObject);
begin Edit3.Clear;Edit4.Clear;
      Memo1.Clear;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;Edit4.Clear;
      Memo1.Clear;Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var i,m,n,p,mzc,bmzc,nzc,bnzc:integer;
begin m:=StrToIntDef(Edit1.Text,1);
      n:=StrToIntDef(Edit2.Text,11);
      If m>n
       then begin p:=m; m:=n; n:=p;
            end;
      Edit1.Text:=IntToStr(m);
      Edit2.Text:=IntToStr(n);
      mzc:=0;
      nzc:=maxint;
      For i:=m to n do
       If (IntToStr(i)[1]>'6')and(i mod 10<4)and(Prost(i))
        then begin p:=ZbirC(i);
                   Memo1.Lines.Add(IntToStr(i));
                   If p>mzc
                    then begin mzc:=p; bmzc:=i;
                               end;
                   If p<nzc
                    then begin nzc:=p; bnzc:=i;
                               end;
            end;
```

```
      Edit3.Text:=IntToStr(bmzc);
      Edit4.Text:=IntToStr(bnzc);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
function TForm1.Prost(b:integer):boolean;
var d:integer;
begin prost:=false;
      If b<4
       then prost:=true
       else If Odd(b)
        then begin d:=3;
                   While (b mod d<>0)and(d<=Sqrt(b)) do d:=d+2;
                   If d>Sqrt(b)
                    then prost:=true;
             end;
end;
function TForm1.ZbirC(b:integer):integer;
var z:integer;
begin z:=0;
      While b>0 do
      begin z:=z+b mod 10;
            b:=b div 10;
      end;
      zbirc:=z;
end;
```

- ***Саставити програм који одређује редни број првог елемента низа задатог формулом $a_n = \frac{x^n}{n!}, n \in N, x \in R$ који је мањи од произвољно малог унетог броја ε.***



Задатак ћемо решити коришћењем две већ познате функције *Fakt* која одређује факторијел броја и *Step* која одређује степен броја (факторијел ће бити реалан број да бисмо могли да радимо са јако великим бројевима). Креираћемо форму као на слици, а затим написати комплетан програм. У процедури ***Edit1KeyPress*** ће можда бити потребно заменити #44, тј. ',' са #46, тј. '.' (у зависности од постављених параметара оперативног система, конкретно, ознаке за децимални зарез).

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then If sender=edit1
             then Edit2.SetFocus
             else Button1.Click
       else If not (key in ['0'..'9',#8,#128])
             then key:=#27
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;Memo1.Clear;
      Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var e,x,a:real;n:integer;
begin x:=StrToFloatDef(Edit1.Text,1); Edit1.Text:=FloatToStr(x);
      e:=StrToFloatDef(Edit2.Text,11);Edit2.Text:=Format('%1.10f',[e]);
      Repeat n:=n+1;
             a:=Step(x,n)/Fakt(n);
             Memo1.Lines.Add(Format('%20.15f',[a]));
      until a>e;
      Edit3.Text:=IntToStr(n);
      Edit1.ReadOnly:=true;Edit2.ReadOnly:=true;
      Button2.SetFocus;
end;
function TForm1.Fakt(b:integer):real;
var f:real;
begin f:=b;
      While b>1 do
      begin b:=b-1;
            f:=f*b;
      end;
      fakt:=f;
end;
```

```
function TForm1.Step(a:real; n:integer):real;
var s:real;
begin s:=a;
      While n>1 do
      begin n:=n-1;
            s:=s*a;
      end;
      step:=s;
end;
```

Уместо ове две функције, згодније је користити једну овакву, која ће смањити проблем великих бројева код факторијела (чиниоци су све мањи бројеви јер *n* стално расте, а *x* је увек исто, па нема проблема са факторијелом када *n* пређе преко 20):

```
function TForm1.Clan(a:real; n:integer):real;
var s:real;
begin s:=a;
      While n>1 do
      begin n:=n-1;
            s:=s*a/n;
      end;
      clan:=s;
end;
```

У том случају у процедури *Button1Click* потребно је наредбу додељивања:

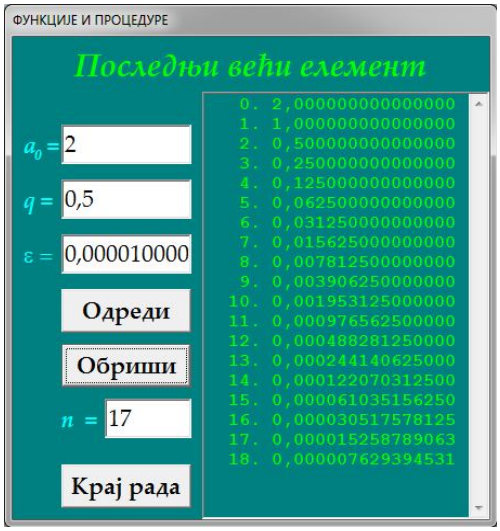
```
a:=Step(x,n)/Fakt(n);
```

заменити једноставнијом наредбом:

```
a:=Clan(x,n);
```

- ***Саставити програм који за унети први члан и количник геометријског низа одређује редни број последњег елемента који је већи од произвољно малог унетог броја ε.***

Да би постојало решење потребно је да a_0 буде веће од ε или да *q* буде веће од 1 (ако је први члан низа мањи од границе и количник је мањи од 1 сви елементи низа биће мањи од границе, па задатак нема решења). Претпоставићемо да ће услови задатка бити испуњени. Задатак ћемо решити коришћењем познате функције *Step* која одређује степен броја. Креираћемо форму као на слици, а затим написати комплетан програм. У процедури ***Edit1KeyPress*** ће можда бити потребно заменити #44, тј. ',' са #46, тј. '.' (у зависности од постављених параметара оперативног система, конкретно, ознаке за децимални зарез). Исписивање елемената низа у мемо поље није обавезно, али је згодно видети понашање низа када се мења почетна вредност броја *x*.




```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then If sender=edit1
             then Edit2.SetFocus
             else If sender=edit2
                   then Edit3.SetFocus
                   else Button1.SetFocus
       else If not (key in ['0'..'9',#8,#44,#128])
             then key:=#27
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.ReadOnly:=false;
      Edit2.Clear;Edit2.ReadOnly:=false;
      Edit3.Clear;Edit3.ReadOnly:=false;
      Edit4.Clear;Memo1.Clear;
      Edit1.SetFocus;
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var e,q,a,c:real;n:integer;
begin a:=StrToFloatDef(Edit1.Text,1) ;Edit1.Text:=FloatToStr(a);
      q:=StrToFloatDef(Edit2.Text,1) ;Edit2.Text:=FloatToStr(q);
      e:=StrToFloatDef(Edit3.Text,11);Edit3.Text:=Format('%1.10f',[e]);
      n:=0;Memo1.Lines.Add('%3d.%18.15f',[n,a]));
      Repeat n:=n+1;c:=a*Step(q,n);
             Memo1.Lines.Add('%3d.%18.15f',[n,c]));
      until c<e;
      Edit4.Text:=IntToStr(n-1);
      Edit1.ReadOnly:=true;
      Edit2.ReadOnly:=true;
      Edit3.ReadOnly:=true;
      Button2.SetFocus;
end;
```

```
function TForm1.Step(a:real; n:integer):real;
var s:real;
begin s:=a;
      While n>1 do
      begin n:=n-1;
            s:=s*a;
      end;
      step:=s;
end;
```

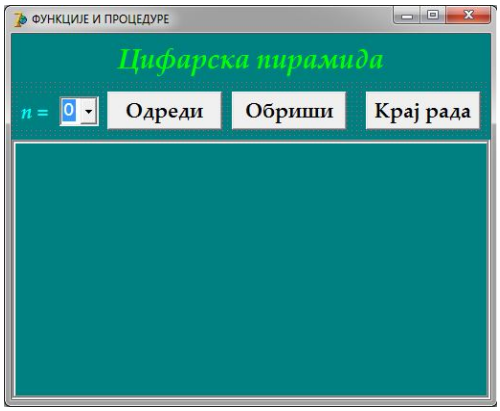
Треба рећи да у овом задатку функција није најбоље решење. Ефикасније и ефектније решење било би са оваквом процедуром:

```
procedure TForm1.Clan(a,q,e:real; VAR n:integer);
begin n:=0;
      While a<=e do
      begin Memo1.Lines.Add(Format('%20.15f',[a]));
            n:=n+1;a:=a*q;
      end;
end;
```

У том случају процедуру која повезује функције са остатком решења је још једноставнија:

```
procedure TForm1.Button1Click(Sender: TObject);
var e,q,a:real;n:integer;
begin a:=StrToFloatDef(Edit1.Text,1); Edit1.Text:=FloatToStr(a);
      q:=StrToFloatDef(Edit2.Text,1); Edit2.Text:=FloatToStr(q);
      e:=StrToFloatDef(Edit3.Text,11);Edit3.Text:=Format('%1.10f',[e]);
      Clan(a,q,e,n);
      Edit4.Text:=IntToStr(n);
      Edit1.ReadOnly:=true;
      Edit2.ReadOnly:=true;
      Edit3.ReadOnly:=true;
      Button2.SetFocus;
end;
```

- ***Саставити програм који исписује цифарску пирамиду.***

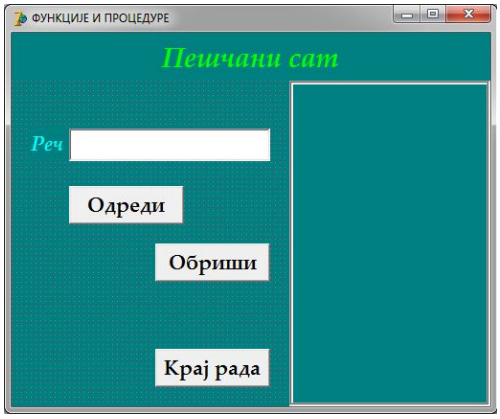


Врло једноставан задатак у коме треба формирати палиндромне речи састављене од онолико различитих цифара колики је редни број слоја пирамиде који се гради. Јасно је да пирамида може имати максимално 10 редова. Да се не би приказао правоугли троугао треба испис центрирати. Коришћењем мемо поља и његове карактеристике *Alignment* може се подесити да се испис центрира. Овде ћемо, међутим, приказати решење где се испис центрира програмски. За испис слоја користићемо функцију *Red* која ће формирати реч од цифара до одговарајућег реда. На врху пирамиде поставићемо цифру 1. Креираћемо форму као на слици, а затим написати комплетан програм.

```
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin ComboBox1.ItemIndex:=9;
          ComboBox1.Enabled:=true;
      Memo1.Clear;
          ComboBox1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,p:integer;
begin p:=ComboBox1.ItemIndex;
      Memo1.Clear;
      For n:=0 to p do
        Memo1.Lines.Add(Red(n,p));
      ComboBox1.Enabled:=false;
      Button2.SetFocus;
end;
function TForm1.Red(n,p:integer):string;
var k:integer;
    a:string;
begin If n=9
       then a:='0'
       else a:=IntToStr(n+1);
      For k:=n downto 1 do
            a:=IntToStr(k)+' '+a+' '+IntToStr(k);
      For k:=1 to p-n do
        a:='  '+a;
      red:=a;
end;
```

Ако желимо да центрирамо садржај мемо поља карактеристиком *Alignment* мемо поља, из функције треба избрисати други циклус који додаје потребан број размака на почетак речи да би била центрирана.

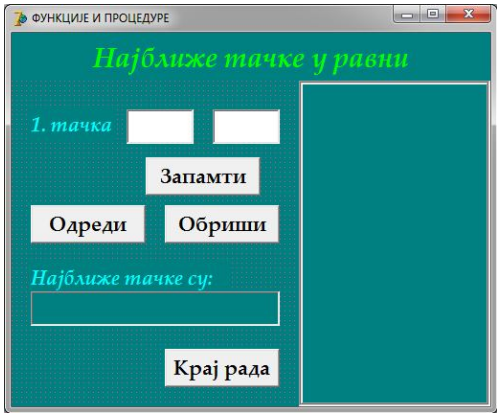
- ***Саставити програм који од унете речи формира пешчани сат тако што у горњем делу одузима, а у доњем делу додаје по једно слово са обе стране речи.***



Једноставан задатак у коме треба формирати све речи које се од почетне добијају одузимањем по једног слова са десне стране и распоређивањем речи у облику два једнакокрака троугла са заједничким врхом. Јасно је да испис мора имати 2*н-1 редова, где је н број слова у унетој речи, зато је форма подешена да реч има максимално 13 слова. Ако желимо да користимо дуже речи морамо проширити мемо поље и повећати висину форме да би се комплетна слика видела. Да се не би приказала два правоугла троугла треба испис центрирати. Коришћењем карактеристике мемо поља *Alignment*, може се подесити да се испис центрира. Овде ћемо, међутим, приказати решење где се испис центрира програмски. За испис слоја користићемо функцију *Red* која ће формирати реч од почетне одузимањем потребног броја слова. Креираћемо форму као на слици, а затим написати комплетан програм.

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then Button1.SetFocus;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Edit1.Clear;Edit1.Enabled:=true;
      Memo1.Clear;
          Edit1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,p:integer;
    a:string;
begin a:=Edit1.Text;
      p:=Length(a);
      Memo1.Clear;
      For n:=1 to (p+1) div 2 do    // горњи део пешчаног сата
        Memo1.Lines.Add(Red(a,n));
      If p mod 2=0      // да се код речи парне дужине средња два слова не би исписивала дупло
       then p:=p-1;
      For n:=p div 2 downto 1 do    // доњи део пешчаног сата је један ред краћи
        Memo1.Lines.Add(Red(a,n));
      Edit1.Enabled:=false;
      Button2.SetFocus;
end;
function TForm1.Red(a:string;n:integer):string;
var k:integer;
    b:string;
begin b:=Copy(a,n,Length(a)-2*n+2);
      For k:=1 to n do
        b:=' '+b;
      red:=b;
end;
```

- ***Саставити програм који одређује које две од четири тачаке задате координатама у равни су најближе.***



Да бисмо решили овај задатак морамо знати како да одредимо растојање између две тачке у кординатној равни. Ако нацртамо у координатној равни две тачке са дужима које представљају њихове апсисе и ординате, приметићемо правоугли троугао чија су два темена на хипотенузи дате тачке. Према томе, растојање између тачака је хипотенуза тог троугла. Катете троугла су разлике апсциса и ордината тачака, па је формула за тражено растојање:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

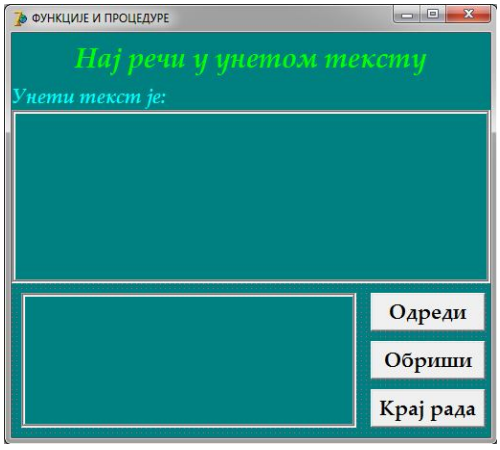
Четири тачке одређују шест дужи. Јасно је да треба одредити међусобна растојања сваке две тачке и, затим најкраће. За унос координата тачака користићемо два едита, зато ће координате бити глобалне променљиве. Упоређиваћемо увек само две дужи, па нећемо имати потребе за превеликим бројем локални променљивих. Да бисмо приказали све тачке, искористићемо мемо поље. Ако има више парова које су на истом, најкраћем растојању приказаћемо све парове. Креираћемо форму као на слици, а затим написати комплетан програм.

```
var
  Form1: TForm1;
  x1,x2,x3,x4,y1,y2,y3,y4:real;
  n:integer=0;
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['-','0'..'9',#8,#13,#44,#128])   // уместо #44 -> , можда треба #46 -> .
               then key:=#27
       else If key=#13
             then If sender=edit1
                        then Edit2.SetFocus
                        else Button1.Click;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Edit1.Clear;Edit1.Enabled:=true;
      Edit2.Clear;Edit2.Enabled:=true;
      Edit3.Clear;
      Memo1.Clear;Memo1.Lines.Add('Тачке су:');
      Edit1.SetFocus;
end;
procedure TForm1.Button2Click(Sender: TObject);
var n,p,q:integer;
    d,dmin:real;
    a,b,c:string;
begin a:=Memo1.Lines.Strings[1];
      n:=Length(a);p:=Pos(' ',a);q:=Pos(';',a);
      y1:=StrToFloat(Copy(a,q+2,n-q));
      x1:=StrToFloat(Copy(a,p+1,q-p-1));
      a:=Memo1.Lines.Strings[2];
      n:=Length(a);p:=Pos(' ',a);q:=Pos(';',a);
      y2:=StrToFloat(Copy(a,q+2,n-q));
      x2:=StrToFloat(Copy(a,p+1,q-p-1));
      a:=Memo1.Lines.Strings[3];
      n:=Length(a);p:=Pos(' ',a);q:=Pos(';',a);
      y3:=StrToFloat(Copy(a,q+2,n-q));
      x3:=StrToFloat(Copy(a,p+1,q-p-1));
      a:=Memo1.Lines.Strings[4];
      n:=Length(a);p:=Pos(' ',a);q:=Pos(';',a);
      y4:=StrToFloat(Copy(a,q+2,n-q));
      x4:=StrToFloat(Copy(a,p+1,q-p-1));
      a:='A12';     // претпостављамо да су најближе прве две тачке, па тражимо ближе од њих
      dmin:=DD(x1,y1,x2,y2);Memo1.Lines.Add('A12: '+FloatToSTr(dmin));
      d:=DD(x1,y1,x3,y3);Memo1.Lines.Add('A13: '+FloatToSTr(d));
      If d<dmin
       then begin dmin:=d;a:='A13';
            end;
      d:=DD(x1,y1,x4,y4);Memo1.Lines.Add('A14: '+FloatToSTr(d));
      If d<dmin
       then begin dmin:=d;a:='A14';
            end;
      d:=DD(x2,y2,x3,y3);Memo1.Lines.Add('A23: '+FloatToSTr(d));
      If d<dmin
       then begin dmin:=d;a:='A23';
            end;
      d:=DD(x2,y2,x4,y4);Memo1.Lines.Add('A24: '+FloatToSTr(d));
      If d<dmin
       then begin dmin:=d;a:='A24';
            end;
      d:=DD(x3,y3,x4,y4);Memo1.Lines.Add('A24: '+FloatToSTr(d));
      If d<dmin
       then begin dmin:=d;a:='A34';
            end;
      b:=FloatToSTr(dmin); // проверавамо да ли има више парова тачака на истом растојању
      c:=Memo1.Lines.Strings[8]; // проверу почињемо од другог пара тачака
      If (b=Copy(c,6,Length(c)-5))and(Copy(c,1,3)<>Copy(a,1,3))
       then a:=a+', '+Copy(c,1,3);
      c:=Memo1.Lines.Strings[9];
      If (b=Copy(c,6,Length(c)-5))and(Copy(c,1,3)<>Copy(a,1,3))
       then a:=a+', '+Copy(c,1,3);
      c:=Memo1.Lines.Strings[10];
      If (b=Copy(c,6,Length(c)-5))and(Copy(c,1,3)<>Copy(a,1,3))
       then a:=a+', '+Copy(c,1,3);
      c:=Memo1.Lines.Strings[11];
      If (b=Copy(c,6,Length(c)-5))and(Copy(c,1,3)<>Copy(a,1,3))
       then a:=a+', '+Copy(c,1,3);
```

```
      c:=Memo1.Lines.Strings[12];
      If (b=Copy(c,6,Length(c)-5))and(Copy(c,1,3)<>Copy(a,1,3))
       then a:=a+', '+Copy(c,1,3);
      Edit3.Text:=a;
      Button3.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var x,y:real;
    a:string;
begin x:=StrToFloatDef(Edit1.Text,1);
      y:=StrToFloatDef(Edit2.Text,1);
      n:=n+1;
      a:='A'+IntToStr(n)+': '+FloatToStr(x)+'; '+FloatToStr(y);
      Memo1.Lines.Add(a);     // унос координата у мемо поље
      If n=4                  // провера да ли су унете координате све четири тачке
       then begin Edit1.Enabled:=false;
                  Edit2.Enabled:=false;
                  Memo1.Lines.Add('');
                  Memo1.Lines.Add('Растојања су:');
                  Button2.SetFocus;
            end
       else begin Edit1.Clear;Edit2.Clear;
                  Edit1.SetFocus;
            end;
end;
function TForm1.DD(x1,y1,x2,y2:real):real;
begin dd:=Sqrt(Sqr(x1-x2)+Sqr(y1-y2));
end;
```

Задатак је мало компликованији, али смо научили да читамо унете податке из мемо поља и да их конвертујемо у одговарајући тип и различите употребе стринг променљивих, па је веома корисно добро га разумети због, евентуалне, касније примене.

- ***Саставити програм који за унети стринг одређује најдужу и најкраћу реч и одређује која од њих се прва појављује.***



Још један задатак где ће стрингови бити у жижи интересовања. Јасно је да је тражење најдуже или најкраће речи исти посао, треба само код услова за дужину речи ставити ознаку > или <. Зато ћемо написати функцију која ће налазити и једну и другу реч. Проблем код тражења речи је што се реч може ограничити размаком или неким од интерпункцијских знакова и да ли ћемо бројеве писане цифрама рачунати као речи или не (на пример: да ли је 2000. година број и реч или две речи и ако се договоримо да су то две речи да ли је 2000 реч или 2000. јер тачка мења смисао броја). Ове проблеме треба да разреши онај ко прави захтев за програмским решењима и ми немамо ништа са тим. Ми само морамо да договорено преточимо у програмски код. У овом случају, претпоставићемо да је унети стринг састављен само од слова и интерпункцијских знакова без цртица за поделу и пренос слогова речи у нови ред. За унос стринга и испис резултата предвидећемо по једно мемо поље. Могуће је уносити текст и ћирилицом и латиницом. Креираћемо форму као на слици, а затим написати комплетан програм.

```
// скуп дозвољених карактера обухвата ћирилична и латинична слова и знаке интерпункције
procedure TForm1.Memo1KeyPress(Sender: TObject; var Key: Char);
begin If not (key in ['a'..'z','A'..'Z','ђ'..'ш','Ђ'..'Ш','.',',',',','!','?',' ',#8,#13,#128])
               then key:=#27
       else If key=#13
             then Button1.Click;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin Application.Terminate;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin Memo1.Clear;Memo1.ReadOnly:=false;
      Memo2.Clear;Memo1.SetFocus;
end;
procedure TForm1.Button1Click(Sender: TObject);
var c,d:integer;
    a,b:string;
begin a:=Memo1.Text+'.';
      If Length(a)>1
       then begin Naj(a,1,b,c);
                  Memo2.Text:='Најдужа реч је '+b;
                  Naj(a,0,b,d);
                  Memo2.Lines.Add('Најкраћа реч је '+b);
```

```
                If c<d
                 then Memo2.Lines.Add('Најдужа је испред најкраће.')
                 else Memo2.Lines.Add('Најкраћа је испред најдуже.');
                Memo1.ReadOnly:=true;
                Button2.SetFocus;
            end;
end;
// параметар с вредностима 1 и 0 одређује да ли се тражи најдужа или најкраћа реч
procedure TForm1.Naj(a:string;s:integer; Var b:string; Var n:integer);
var c:string;
    g:integer;
begin a:=Obrisi(a);      // брише водеће размаке и знакове интерпункције
      g:=Duzina(a);      // броји слова у првој речи унутар текста
      n:=g-1;            // прва реч се проглашава најдужом или најкраћом
      b:=Copy(a,1,n);    // памти се вредност прве речи
      Delete(a,1,g);     // брише се прва реч из унетог текста
      a:=Obrisi(a);      // брише водеће размаке и знакове интерпункције у новом тексту
      While Length(a)>0 do     // тражи се дужа или краћа реч све док има слова у тексту
      begin g:=Duzina(a);      // броји слова у првој речи унутар текста
            c:=Copy(a,1,g-1); // памти се вредност нове речи
            Delete(a,1,g);    // брише се реч из текста
            If (Length(c)>n)and(s=1)      // проверава да ли је нова реч најдужа
             or(Length(c)<n)and(s=0)      // проверава да ли је нова реч најкраћа
               then begin n:=g-1;         // памти се нова најдужа или најкраћа дужина
                          b:=c;           // памти се вредност речи
                    end;
            a:=Obrisi(a);     // брише водеће размаке и знакове интерпункције
      end;
end;
//функција одређује број слова у следећој речи у унетом тексту
function TForm1.Duzina(a:string):integer;
var g:integer;
begin g:=Pos(' ',a);
      If g=0 then g:=Length(a);
      If (Pos(',',a)<g)and(Pos(',',a)>0)
       then g:=Pos(',',a);
      If (Pos('.',a)<g)and(Pos('.',a)>0)
       then g:=Pos('.',a);
      If (Pos('!',a)<g)and(Pos('!',a)>0)
       then g:=Pos('!',a);
      If (Pos('?',a)<g)and(Pos('?',a)>0)
       then g:=Pos('?',a);
      duzina:=g;
end;
// функција брише дупле размаке и удвојене знаке интерпункције
function TForm1.Obrisi(a:string):string;
begin While (Length(a)>0) and
            (not (a[1] in ['a'..'z','A'..'Z','ђ'..'ш','Ђ'..'Ш'])) do
        Delete(a,1,1);
      obrisi:=a;
end;
```

Задаци за самостални рад

- ***Саставити програм који израчунава обим и површину троугла задатог помоћу координата његових темена.***
- ***Саставити програм који израчунава вредност функције задате формулом:***

$$f(x)=\begin{cases}\min(x,-5), & x\le 0\\ 15\cdot x-5, & 0<x<1\\ \max(x,10), & x\ge 1\end{cases}$$

- ***Саставити програм који израчунава вредност функције задате формулом:***

$$f(x)=1+\frac{1}{1+\frac{1}{2}}+\frac{1}{1+\frac{1}{2}+\frac{1}{3}}+\cdots+\frac{1}{1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{x}},\quad x\in N$$

- ***Саставити програм који исписује све бројеве од m до n који су дељиви сваком својом цифром.***
- ***Саставити програм који симулира бацање коцкице све док се три пута за редом не добије исти број, а затим исписати статистику појављивања сваког броја.***
- ***Саставити програм који исписује назив дана за било који датум за године од 1900. до 2100.***
- ***Саставити програм који исписује календар за било који месец за године од 1900. до 2100.***
- ***Саставити програм који одређује разлику у данима између два датума за године од 1900. до 2100.***

- ***Саставити програм који за унети датум одређује датум који је био пре и који ће бити после унетог броја дана.***
- ***Саставити програм који одређује колико има бројева од т до п који нису прости и чија је последња цифра мања од 6, а прва већа од 4, исписати их одредити збир њихових цифара.***
- ***Дати су бројеви: a, b, c, d. Формирају се два низа на следећи начин:***
  први низ: a, a+b, a+b+a, a+b+a+b, a+b+a+b+a, ...
  други низ: c, c+d, c+d+c, c+d+c+d, c+d+c+d+c, ...
  Саставити програм који одређује првих п елемената низова који су једнаки и исписати њихове редне бројеве.
- ***Саставити програм који одредује збир непарних цифара бројева од м до н.***
- ***Саставити програм који исписује двоцифрене бројеве код којих је цифра десетица дељива цифром јединица.***
- ***Саставити програм који одредује колико пута се цифра ц појављује у бројевима од м до н.***
- ***Саставити програм који одредује број од м до н који има највише делилаца.***
- ***Саставити програм који исписује све двоцифрене бројеве чија се сума цифара не мења при множењу бројевима од 2 до 9.***
- ***Саставити програм који исписује све троцифрене бројеве код којих је сума цифара једнака датом целом броју.***
- ***Саставити програм који одређује које две од четири тачаке задате координатама у простору су најближе.***
- ***Саставити програм који за два уписана стринга одређује најдужи заједнички подниз***
- ***Написати програм којим се одређују сви троцифрени бројеви који су једнаки суми факторијела својих цифара.***
- ***Кинески цар Циу Ши је развијао своје царство и свако село је било у обавези да му у току године пошаље лопту од ћилибарда одређеног полупречника. Он је лопте слагао на под у једној од својих одаја, тако да чине троугао једнаких страница.Када би освајао села или оснивао нова увек је водио рачуна о броју и о томе да када те године добије по лопту од свих села може да их дода на сваку страницу по један ред и да опет добије троугао једнаких страница. Написати програм у коме се задаје тренутан број лопти Н на једној страници троугла, а који израчунава колико села укупно цар мора да има те године да би допунио свој троугао до новог троугла.***