Програмирање
(предавања)

Час 1. Програмирање, програмски јазик, Visual Basic, основни концепти на програмирање

Програмирање

- предмет: Програмирање (летен семестар 2012/2013)
- наставник: доц. д-р Ирена Стојковска
- часови: 4 часа неделно
- термини: среда 12-14 и четврток 10-12

Литература

- предавања
- вежби
- материјали кои ќе бидат објавени на страната на предметот http://www.institutzamatematika.com/index.php/KOMPJUTERSKI_PRAKTIKUM_2

Начин на полагање

- редовност
- 2 колоквиуми
- домашна работа и/или проект (презентација)

Досегашни статистики

- околу 90% од студентите го положуваат предметот при неговото запишување по прв пат, околу 90% од нив преку колоквиуми
- околу 10% од студентите го презапишуваат предметот по втор пат, околу 90% од нив не доаѓале на часовите

Прашања?

Зошто програмирање?

- компјутерот е машина, оперира врз база на команди
- **програмирање** е давање инструкции на компјутерот да сработи некоја работа
- основен аспект на компјутерите: работи тоа што ќе му се каже да го работи и го сработува побрзо отколку што може човекот

Како функционира програмирањето?

- **програма** е збир од детални инструкции кои му кажуваат на комјутерот што точно да работи
- програмите се пишуваат во **програмски јазик**

  - машински јазик (јазикот со кој „зборува“ компјутерот, 1=„on“ и 0=„off“)

  - low level јазици (покрај бинарен код содржат и симболичен код, Assembler, BAL, …)

  - high level јазици (код близок на англискиот јазик, Pascal, FORTRAN, COBOL, C, C++, Prolog, Java, **Visual Basic**, …)

Како функционира програмирањето?

- **компајлер** е специјална програма која ги конвертира инструкциите напишани на програмскиот јазик во машински јазик
- да се биде **програмер** не треба да се знае што работи комјутерот ниту пак како работи, туку треба да се разбере како работи програмскиот јазик

Visual Basic

- **"visual"** се однесува на методот кој се користи при креирање тоа што го гледа корисникот
- **"basic"** се однесува на програмслиот јазик BASIC (Beginner's All-purpose Symbolic Instruction Code), најкористениот програмски јазик во историјата на комјутерите
- **Visual Basic (VB)** е Windows апликација

Visual Basic

- VB е моќна алатка која најчесто програмерите ја користат за пишување, тестирање и извршување на **Windows апликации** (интерактивен модел, колекција од еден или повеќе фајлови компајлирани во една извршна програма)
- програмата напишана во VB се нарекува **VB-проект** (колекција од фајлови)
- **(Пример 1. VB-проект: Пресметување на плоштина и волумен на коцка)**

Основни концепти на програмирањето

Пет чекори до создавање на **ефикасна програма**

- **анализа на проблемот** (разбирање на проблемот, што да се очекува од проблемот, што треба програмата да направи, начини на решавање на проблемот и врската меѓу влезот и очекуваниот излез)
- **дизајнирање** (планирање на решението на проблемот, наоѓање на логична низа од прецизни инструкции кои го решаваат проблемот - **алгоритам**)

Основни концепти на програмирањето

- **кодирање** (преведување на алгоритмот во програмски јазик преку пишување на програма)
- **тестирање и откривање на грешките** (лоцирање и отстранување на грешките (bug) во програмата ако постојат, проверка дали програмата работи онака како што е очекувано да работи)
- **документација** (организирање на целокупниот материјал кој ја опишува програмата, коментари во кодот, детален опис за што служи програмата и како да се користи истата)

Принципи на добар стил на програмирање

- конвенции на именување на променливи, процедури, функции
- именување на фајловите и нивна организација во директориуми
- форматирање и вовлекување (indentation) на кодот
- коментари и документација
- намалување на бројот на грешки преку тестирање
- **(илустрација на петте чекори при програмирањето на Пример 1.)**

Заклучок

- комјутерот е машина, работи тоа што ќе му се каже да работи, и го сработува побрзо отколку човекот
- програмирањето е пишување на програма во програмски јазик која содржи детални инструкции со кои му се кажува на компјутерот што да работи
- програмскиот јазик Visual Basic во основа го има најкористениот програмски јазик BASIC кој е лесен за разбирање, програмирањето е раководено од настани, што го прави привлечен за креирање на Windows апликации

Заклучок

- следење на основните концепти на програмирањето + разбирање на програмскиот јазик Visual Basic + добар стил на програмирање = успешно решена задача (проблем)

Програмирање
(предавања)

Час 2. Алгоритми

Од претходниот час ...

- програмирањето е пишување на програма во програмски јазик која содржи детални инструкции со кои му се кажува на компјутерот што да работи
- еден од основните концепти на програмирањето е дизајнирање т.е. планирање на решението на проблемот, наоѓање на логична низа од прецизни инструкции кои го решаваат проблемот - **алгоритам**

Главни карактеристики на алгоритмите

- **алгоритам** е подредена низа од прецизни, добро дефинирани инструкции, која извршува одредена задача и запира по конечно многу време
- главни карактеристики на алгоритмите:

  - алгоритамот мора да има почеток и крај

  - обсегот на влезни податоци за кои алгоритамот работи треба да биде внимателно одреден

  - еден ист алгоритам може да биде прикажан на различни начини

Главни карактеристики на алгоритмите

- може да постојат неколку алгоритми кои решаваат еден ист проблем

- алгоритмите за еден ист проблем може да бидат базирани на различни идеи и може да го решаваат проблемот со значајно различни брзини

- алгоритамот треба да запре по разумно многу време

Алгоритамско решение на еден проблем

- Не секој проблем има „добро“ алгоритамско решение. На пример, постојат

  - **нерешливи проблеми** (не постои алгоритам кој го решава проблемот - Halting Problem: За даден проблем и дадени влезни податоци, да се одреди дали алгоритмот запира по конечно многу време или работи бесконечно.)

  - **„тешки“ проблеми** (на алгоритмот му треба многу време за да го реши проблемот - Traveling Salesmen Problem)

  - **проблеми за кои се уште не е најдено алгоритамско решение** (отворени проблеми - K-server problem)

Фази во дизајнирањето на алгоритмите

- **дефиниција на проблемот** (разбирање на проблемот, разгледување на сите можни случаи на проблемот)
- **одредување на видот на пресметките**
- **одредување на методот** кој се применува за решавање на проблемот (точно или приближно решение)
- **дизајнирање на алгоритмот** (блок дијаграм или псевдо код)

Фази во дизајнирањето на алгоритмите

- **евалуација на алгоритмот** (тестирање на алгоритмот)
- **докажување на исправноста на алгоритмот** (алгоритамот го дава бараниот резултат за секој исправен влез по конечно многу време)
- **анализа на алгоритмот** (се проверува временската и просторната ефикасност на алгоритмот)
- **кодирање на алгоритмот** (предизвик за составување на програма, и можност за добивање на решението на поставениот проблем)

Три категории на алгоритамски операции

- **секвенционални операции** (инструкциите се извршуваат по ред)
- **условни операции** (се поставува да/не прашање и се избира следната инструкција врз база на одговорот)
- **итеративни операции (loops)** (се повторува извршувањето на блок од инструкции)

Блок дијаграм

Симболи кои се користат за блок дијаграмот

- поврзувачка насочена линија
- терминал (почеток или крај)
- влез/излез
- процесирање (аритметички операции и работа со податоци)
- одлука (еден влез и два излеза - „да“ или „не“)
- поврзувач на различни насочени линии

- **(пример за блок дијаграм за проблемот од Пример 1. P и V на коцка - исто така е пример за секвенционални операции)**

Псевдо код

- **псевдо код** е скратена верзија на евентуалниот компјутерски код, му овозможува на програмерот повеќе да се фокусира на чекорите за решавање на проблемот, отколку како да го искористи програмскиот јазик
- Предности на псевдо кодот

  - покомпактен од блок дијаграмот

  - полесно се трансформира во програмскиот јазик
- **(пример за псевдо код за проблемот од Пример 1. P и V на коцка - исто така е пример за секвенционални операции)**

Типови на алгоритми

- **нерекурзивни алгоритми** (не се повикуваат на ист алгоритам или функција)
- **рекурзивни алгоритми** (користат иста функција повикувајќи се себе си)
- **(пример за нерекурзивен и рекурзивен алгоритам за Пример 2. Генерирање на првите 30 Фибоначиеви броеви - исто така е пример за итеративни операции)**

Програмирање
(предавања)

Час 3. Основи на програмирањето во Visual Basic

Од претходниот час ...

- алгоритмот е конечна низа од прецизни инструкции која нуди начин на решавање на проблемот за разумно многу време
- кодирањето на алгоритмот е пишување на програма во програмскиот јазик - предизвик и можност за добивање на решението на проблемот
- познавањето на програмскиот јазик е важна компонента за успешно решавање на проблемот

Основни типови на податоци

- Двата **основни типа на податоци** кои може да ги обработува Visual Basic се:

  - броеви

  - текстуални низи (Strings)

- Податоците се складираат во:

  - константи (не ја менуваат својата вредност, можност за декларирање на нови константи со **Const** *constname =expression,* каде *expression* се состои од константи, аритметички или логички операции)

  - променливи (може да ја менуваат својата вредност)

- (Илустрација на декларирањето на нови константи во процедурата **Form_Activate**)

Променливи

- **Променливата** е име кое се користи за да пристапиме до содржината на податокот.

  - **името на променливата** може да биде составено најмалку од 1 и најмногу од 255 карактери,

  - мора да започне со буква,

  - може да содржи букви, цифри и подвлечени линии ("_"),

  - Visual Basic не прави разлика меѓу малите и големите букви во името на променливата (имињата brojnacifri и brojNaCifri се однесуваат на една иста променлива)

Променливи

- доделувањето на вредност на променлива (со "=“)

1) *var = num* (на променливата *var* и се доделува нумеричката вредност *num*, пр. *a=3*)

2) *var = str* (на променливата *var* и се доделува текстуалната вредност *str*, пр. *ime="Ana"*)

3) *var2 = var1* (на променливата *var2* и се доделува вредноста од променливата *var1*, пр. *a=3:b=a*, резултира да променливата *b* добие вредност *3*)

4) *var = expression* (прво се пресметува вредноста на изразот *expression*, а потоа добиената вредност се доделува на променливата *var*, пр. *a=3:b=a^2*, резултира да променливата *b* добие вредност *9*)

- (Илустрација на доделувањето на вредност на променлива во процедурата **Form_Activate**)

Декларирање на променливи

Dim *var* **As** *type*

- *var* е **име на променливата**
- *type* е **типот на променливата**
- се декларира на почеток од процедурата, или во делот **General Declarations** за да важи за сите процедури од формата
- декларирањето се препорачува за поголема ефикасност на програмата (променливите зафаќаат помалку место од меморијата)
- ако некоја променлива не е декларирана се зема дека е од типот **Variant**

Типови на променливи (*type*)

Integer - цели броеви од -32768 до 32767

Long - цели броеви од -2147483648 до 2147483647

Single - бројот 0, броевите од $1.40129 \cdot 10^{-45}$ до $3.40283 \cdot 10^{38}$ со најмногу 7 значајни цифри и негативите на овие броеви

Double - бројот 0, броевите од $4.9406520 \cdot 10^{-324}$ до $1.7976931348623 16 \cdot 10^{308}$ со најмногу 17 значајни цифри и негативите на овие броеви

String - текстови од 0 до 32767 знака (било кој знак, дури и празно место)

Boolean - две вредности True или False

Currency - броеви од -922337203658477.5808 до 922337203658477.5807 со најмногу 4 дец. места

Date - броеви кои претстав. датуми од 1 јан.100 до 31 дек. 9999

Byte - броеви од 0 до 255

Variant - сите типови на податоци

Аритметички операции и операции со стрингови

- **Основни операции**
  + (собирање)
  - (одземање)
  * (множење)
  / (делење)
- **Останати операции**
  ^ (степенување)
  \ (целобројно делење)
  Mod (остаток при делење)
  & (слепување стрингови)

(Тестирање на операциите во **Immediate Window**)

Приоритет на извршување на аритметичките операции

( ) - вредноста во загради прва се пресметува

^ - дигањето на степен е второ по приоритет

- - создавањето на негативен број е трето

* / - множењето и делењето се четврти

\ - целобројното делење е петто

Mod - остатокот при делење е шести

+ - - собирањето и одземањето се последни

- (Тестирање на приоритетот на извршување на аритметичките опеарации во **Immediate Window**)

Логички релации и логички операции

- **Логички релации**

  = (еднакво на)

  <> (различно од)

  < (помало од)

  > (поголемо од)

  <= (помало или еднакво на)

  >= (поголемо или еднакво на)

- **Логички операции**

  Not (не)

  And (и)

  Or (или)

  Xor (или...или)

(Тестирање на логичките релации и операци во **Immediate Window**)

Нумерички функции

- Visual Basic има повеќе предефинирани функции (built-in functions).
- Секоја функција е поврзана со најмалку една **влезна вредност** и враќа една **излезна вредност**.
- Влезните вредности и излезната вредност може да се или нумерички или текстиални.
- **Нумерички функции** се оние кои имаат нумерички влез и излез.
- Нумерички функции во Visual Basic: **Abs, Atn, Cos, Exp, Fix, Int, Log, Rnd, Round, Sgn, Sin, Sqr, Tan**
- (Тестирање на нумеричките функции во **Immediate Window**)

Влез / Излез

- влез преку **InputBox функција** (текстуални или нумерички вредности)

var=**InputBox**(*prompt*) или *var*=**Val**(**InputBox**(*prompt*))

- излез со **Print метод**

object.**Print** [*outputlist*]

- излезот се уредува со функциите **Spc(n)** и **Tab(n)** и сибмолите **,** (запирка) и **;** (точка запирка)

- излез со **MsgBox функција** (за да се привлече вниманието на корисникот)

MsgBox(*prompt*)

- (Тестирање на излезот со Print методот во процедурата **Form_Activate**)

Условна структура If…Then

- прост облик

If *cond* **Then** *statement1* **Else** *statement2*

- сложен облик

If *cond1* **Then**
statements1
ElseIf *cond2* **Then**
statements2
…
ElseIf *cond(n)* **Then**
statements(n)
Else
statements(n+1)
End If

- избегнување на **вгнездени** условни структури кога условите се независни

If *cond1* **Then**
If *cond2* **Then**
statements
End If
End If

може да се презапише како

If *cond1* **And** *cond2* **Then**
statements
End If

(Тестирање на сложениот облик во процедурата **Form_Activate**)

Условна структура Select…Case

- уште еден начин на избегнување на вгнездените If…Then
- изборите зависат од вредноста на **селекторот** (*selector*) кој може да биде нумерички или текстуален израз
- **листата вредности** (*valuelist*) која го одредува изборот може да содржи една или повеќе вредности одделени со запирка (пр. 3,4,5 или 3 **To** 5 или **Is** <= 5)

```
Select Case selector
    Case valuelist1
        statements1
    …
    Case valuelist(n)
        statements(n)
    Case Else
        statements(n+1)
End Select
```

Циклус For…Next

- кога се знае точно колку пати треба да се повтори една група од наредби

For *counter=start* **To** *end* [**Step** *step*]

statements

Next *counter*

- бројачот (*counter*) е некоја нумеричка променлива, *start* е почетната вредност, а *end* е крајната вредност на бројачот, *step* е показател за колку бројачот се зголемува при секое повторување на циклусот (може да биде и негативен)

- избегнувајте промена на вредноста на бројачот внатре во циклусот, може да доведе до бескраен циклус

- дозволено е вгнездување на друг For…Next циклус, но не смее да има преклопувања и бројачот треба да е различен

- предвремен излез од циклусот со **Exit For**

Циклус Do…Loop

- повторува група од наредби се додека е точен некој услов или се додека не стане точен некој услов
- постојат четири облика на Do…Loop циклусот

1) **Do While** *cond*
   statements
   Loop
2) **Do**
   statements
   Loop While *cond*
3) **Do Until** *cond*
   statements
   Loop
4) **Do**
   statements
   Loop Until *cond*

- предвремен излез од циклусот со **Exit Do**
- дозволено е вгнездување на друг циклус Do…Loop
- прекинување на бескраен циклус со **Ctrl**+**Break** или **Break** копчето од лентата со инструменти

(**Пример 3. Пресметај го збирот на првите *n* природни броеви.**)

Грешки при програмирањето

- синтаксички грешки (се препознаваат од smart editor кога се внесуваат)

  пр. Primt, 2ka=2, a=b+
- грешки за време на извршување на програмата (run-time errors) - може да настане заради неправилна синтакса или од неможност на компјутерот да изврши одредена задача

  пр. varInv=1/var (ако вредноста на var е 0)
- логички грешки (кога програмата не работи онака како се очекува да работи)

  пр. av = min + max / 2
- справување со грешките **On Error Go To** *line*

Следува ...

- во Основите на програмирањето во Visual Basic спаѓаат уште и:
  - **Текстуални низи (Strings)**
  - **Низи од променливи (Arrays)**
- пред графика и работа со датотеки:
  - **Основи на визуелното програмирање во Visual Basic**
- последни за овој семестар ќе бидат основите на:
  - **Графика во Visual Basic**
  - **Работа со датотеки во Visual Basic**

Програмирање
(предавања)

Час 4. Текстуални низи (Strings)

Од претходниот час ...

- двата основни типа на податоци кои може да ги обработува Visual Basic се броеви и текстуални низи (Strings)
- податоците се складираат во

  - константи (**Const** *constname =expression*) и

  - променливи (**Dim** *var* **As** *type*)
- типот на променливи резервиран за текстуални низи е String со кој се складираат текстови од 0 до 32767 знака (било кој знак, дури и празно место)
- операцијата за стрингови е слепување (&)

Стринг константи и стринг изрази

- **стринг константа (стринг)** е низа од карактери (текстуална низа) која се третита како една целина

  - примери за стрингови се: реченици, зборови, букви, имиња, телефонски броеви, адреси, матични броеви
- **стринг израз** е комбинацијата од стрингови и знаци за спојување (&) која дава резултат еден стринг
- **нулти стринг** или **празен стринг** ("") - обично е почетна вредност за стринг променлива, како 0 кај нумеричките променливи

ASCII карактерно множество (0-255)

- функции **Asc** (го дава ASCII кодот на карактерот), **Chr** (го дава карактерот кој одговара на ASCII кодот)
- корисни ASCII кодови: 48-57 (цифри 0-9), 65-90 (големи букви), 97-122 (мали букви), 32 (празно место), 34 (наводници)
- споредување на стрингови со логичките релации =, <, >, <=, >=, <> (се споредуваат ASCII кодовите на карактерите првиот, вториот, итн.)
- (Тестирање на ASCII кодовите и споредувањата во **Immediate Window**)

Функции поврзани со стрингови

- Стринг функции

  - Left, Mid, Right (издвојување карактери како дел од стингот)

  - UCase, LCase (претварање во големи, односно мали букви)

  - Trim, LTrim, RTrim (отстранување на празни места)

  - String (некој карактер повторен неколку пати)

  - Str (претварање во стринг)

  - Format (форматирање на излезот)

Функции поврзани со стрингови

- Нумерички функции со стринг аргументи
  - **Len** (должина на стринг)
  - **InStr** (позиција на подстринг во стринг)
- (Тестирање на функциите поврзани со стрингови во **Immediate Window**)

- (**Пример 4. Одреди колку пати дадена буква се среќава во даден збор.**)
- (**Пример 5. Одреди колку збора има во една реченица.**)

Програмирање
(предавања)

Час 5. Низи од променливи (Arrays)

Од претходниот час ...

- променлива (или проста променлива) е име кое се користи за да пристапиме до содржината на податокот
- со **Dim** *var* **As** *type* се декларира типот на променливата (*type* = Integer, Long, Single, Double, String Boolean, Currency, Date, Byte, Variant)
- името на променливата мора да започне со буква, и може да содржи само букви, бројки и подвлечени линии ("_")

Низа од променливи - дефиниција, декларирање

- **низа од променливи** е колекција од прости променливи од ист тип
- се декларира:

  - името на низата,

  - типот на променливи,

  - димензијата и

  - бројот на променливи по секоја димензија
- декларирање на низа од променливи со фиксна големина:

  Dim *arrayName*(n) **As** *varType*

  Dim *arrayName*(m To n) **As** *varitype* (за m<=n)

Низа од променливи - дефиниција, декларирање

- **Пример.** (еднодимензионални)

  Dim A(3) **As** Integer, ги содржи Integer променливите A(0), A(1), A(2), A(3)

  Dim A(1 To 3) **As** Integer, ги содржи Integer променливите A(1), A(2), A(3)

- **Пример.** (дводимензионални)

  Dim B(1 To 3, 2 To 4) **As** Integer, ги содржи Integer променливите B(1,2), B(1,3), B(1,4),B(2,2), B(2,3), B(2,4), B(3,2), B(3,3), B(3,4)

Низа од променливи - дефиниција, декларирање

- декларирање на низа од променливи со променлива големина (**динамички низи од променливи**):

  1) обезбедуваме место во меморијата декларирајќи го името и типот на променливи:

  Dim *arrayName*() **As** *varType*

  2) и кога ќе имаме информација за димензијата на низата и бројот на променливи по секоја димензија:

  ReDim *arrayName*(m1 To n1, …, mk To nk) **As** *varType*

  3) а доколку сакаме да ни се задржат веќе доделените вредности:

  ReDim Preserve *arrayName*(m1 To n1, …, mk To nk) **As** *varType*

Употреба на низите од променливи

- при работа со поголем број на еднородни податоци (листа со оценки, список на студенти, низа броеви)
- лесен пристап кон секој од податоците (преку индексите на елеменетите во низата)
- се користат за задачи од типот на сортирање, подредување, операции со матрици

Примери

- **Пример 7. Пресметај го просекот од оценките добиени во првата година од студирањето.**
- **Пример 8. Најди ги најголемиот и најмалиот број меѓу n дадени броеви.**
- **Пример 9. Пресметај ја нормата на матрицата , ако**

$$\| A \| = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^{2} \right)^{1/2}$$

Програмирање
(предавања)

Час 6. Основи на вузуелното програмирање во Visual Basic

Визуелното во Visual Basic

- Програмите во Visual Basic прикажуваат прозорци (наречени **форми**) со полиња каде корисникот внесува (или менува) одредени информации и копчиња кои се кликаат за да се изврши некоја команда.
- Полињата и копчињата се наречени **контроли**.
- Формите и контролите се викаат **објекти**.

Контроли во Visual Basic

- Секоја од иконите во кутијата со инструменти (Toolbox) претставува одредена контрола која може да биде нанесена на формата. Основни (најчесто користени) контроли:

  - **текстуално поле** (Text Box) - се користи за добивање на информации од корисникот,

  - **етикета** (Label) - етикетата се поставува од левата страна на текстуалното поле за да му објасни на корисникот која информација да ја внесе во текстуалното поле, но етикетите може да се користат и за прикажување на одреден излез,

  - **командно копче** (Command Button) - корисникот клика на командното копче за да иницира одредено дејствие,

  - **поле за слика** (Picture Box) - се користи за прикажување на текстуален или графички излез.

Својства на објектите

- Во прозорецот на својствата (Properties Window) се прикажани **својствата** и нивните **почетни (иницирачки) вредности** за селектираниот објект
- Најчесто користени својства
  - за TextBox се: Text, Font, Enabled, Visible, …
  - за Label се: Caption, Font, Alignment, Visible, …
  - за Command Button се: Caption, Font, Enabled, Visible, …
  - за Picture Box се: Picture, Font, Visible, …
- Програмите во Visual Basic се состојат од три дела: **визуелен приказ (interface), вредности на својствата и код**.
- Вредности на својствата на обектите може да се доделуваат и преку кодот.

Имиња на објектите

- Почетни имиња кои се доделени на објектите се од обликот Text1, Text2, …, Label1, Label2, …, Command1, Command2, …, Picture1, Pictire2,…
- Името на објектот може да се промени со менување на вредноста на **својството Name** кое секогаш се наоѓа на почетокот од листата со својства на селектираниот објект.
- Според конвенциите за именување на објектите името на објектот би требало да започнува со префикс од три мали букви кој го определува објектот, така
  - за Text Box се користи префикс **txt**, на пример txtPrezime
  - за Label се користи префикс **lbl**, на пример lblNaslov
  - за Command Button се користи префикс **cmd**, на пример cmdPresmetaj
  - за Pictute Box се користи префикс **pic**, на пример picPrikaz
  - за Form се користи префикс **frm**, на пример frmVlez

Настани во Visual Basic

- Кога се извршува една Visual Basic програма, на екранот се појавуваат формата и контролите на неа. Ништо не се сучува се додека корисникот не превземе одредено дејствие, како на пример кликање на некоја конрола. Таквото дејство се нарекува **настан**.
- Три чекори во креирањето на Visual Basic пороограма:

  - креирање на визуелниот изглед (interface): нанесување на објектите на формата,

  - доделување на почетни вредности на својствата на секој од објектите,

  - пишување на кодот кој треба да се извршува кога ќе се повика одреден настан.

Кодот во Visual Basic

- **Кодот** се состои од реченици (програмски редови) кои извршуваат прецизни задачи за време на извршувањето на програмата.
- **Вредностите на својствата** на еден објект може да се променат и во кодот со реченици од обликот

  objectName.property = setting

  На пример,

  txtPrezime.Text = "" или cmdPresmetaj.Visible = True
- Преку кодот може да се повикаат одредени **методи** - дејствиа кои соодветниот објект може да ги извршува

  objectName.method [*display*]

  На пример,

  txtPrezime.SetFocus или picPrikaz.Print "Rezultatite se:"

Кодот во Visual Basic

- Кодот кој треба да се извршува со повикување на некој настан се пишува во рамките на **процедурата на настанот**.

  Private Sub *objectName_event*()

  statement(*s*)

  End Sub

  (Зборот Private означува дека соодветната процедура на настанот не може да се повика од настан од друга форма, за таа цел се користи јавна процедура - Public Sub)

Процедури во Visual Basic

- **Структурното програмирање** подразбира дека задачата (проблемот) е поделен на помали подзадачи (подпробеми) кои се решаваат еден по еден. За таа цел Visual Basic располага со две алатки:

  - Sub процедури и

  - Function процедури.

  Тие се наречени уште и **генерални прцедури** чија цел е да го елиминираат повторувањето на кодот и може да бидат искористени и во други програми.
- **Sub процедура** е дел од програмата која извршува една или повеќе задачи, има свое име, и напишана е во облик на одделен дел од програмата.

```
Private Sub ProcedureName ()
        statement(s)
End Sub
```

Процедури во Visual Basic

- Sub процедурата се повикува со реченицата

  Call *ProcedureName*

- По правило, една Sub процедура треба да извршува само една задача или неколку сродни задачи, и треба да биде што е можно пократка.
- Една Sub процедура може да повика друга Sub процедура преку нејзиниот код.
- За пренесување на вредностите на променливите од една во друга Sub процедура потребно е соодветните променливи да бидат декларирани во делот General Declarations (доколку VB-проектот се состои од една форма) или во стандареден модул како глобални променливи (доколку VB-проектот се состои од повеќе од една форма).

Процедури во Visual Basic

- Покрај вградените функции, може да се дефинираат и нови функции наречени **Functon процедури** или кориснички дефинирани функции.

  Private Function *FunctionName*(*var1* As *Type1, var2* As *Type2, …*) As *dataType*

  statement(*s*)

  FunctionName = expression

  End Function

- Function процедурите имаат единствена излезна вредност која може да биде нумеричка или текстуална. Function процедурите може да бидат искористени во изрази на ист начин како и вградените финкции.

Процедури во Visual Basic

- Function процедурите се разликуваат од Sub процедурите според начинот на кој се пристапува до нив. Функциите се повикуваат со нивно поставување на местата на кои би се очекувало да се појави константа, променлива или израз. За разлика од Function процедурите, Sub процедурите не може да бидат вметнати во некој израз.
- Преку една Function процедура може да се повика друга Function или Sub процедура.

(Пример 10. Табелирање на функција.)

(Пример 11. Множење на матрици.)

Програмирање
(предавања)

Час 7. Графика во Visual Basic
(основни поими)

Основни чекори при графички прикази во Visual Basic

- задавање на координатниот систем
- користење на графички методи за цртање на линии, правоаголници, точки, кругови (Line, PSet, Circle)
- поставување на текст на соодветните места (својства: CurrentX, CurrentY, методи: Print, TextHeight, TextWidth)

Задавање на координатниот систем

- Основното поставување (default) на координатниот систем е со координатниот почеток (0,0) во горниот лев агол, x-оската е насочена од лево на десно, y-оската е насочена од горе на долу.
- Со менување на вредностите на **својствата ScaleWidth, ScaleHeight, ScaleLeft, ScaleTop** се поставува нов координатен систем. На пример, со

```
Picture1.ScaleWidth = 30
Picture1.ScaleHeight = - 30
Picture1.ScaleTop = 15
Picture1.ScaleLeft = -15
```

поставен е координатен систем со 30 единици по ширина и висина, и координатен почеток на средината на Picture1, при што x-оската е насочена од лево на десно, а y-оската е насочена од долу на горе (насоката се менува со знакот -).

Задавање на координатниот систем

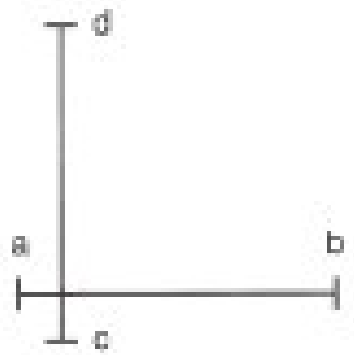
- Истото прилагодување на кординатниот систем се постигнува со **Scale методот**:

  Picture1.Scale (-15, 15) - (15, -15)

  или во општ случај

  Picture1.Scale (a, d) - (b, c)

  за поставување на координатен систем од a до b по x-оската и од c до d по y-оската.



Графички методи

- Отсечка со почеток во точката (x1,y1) и крај во точката (x2,y2) се црта со **методот Line** како

  Picture1.Line (x1,y1) - (x2,y2)

- Точка со координати (x,y) се нанесува со **методот PSet** како

  Picture1.PSet (x,y)

- Кружница со центар во (x,y) и радиус r се црта со **методот Circle** како

  Picture1.Circle (x,y), r

Графички методи

Посложена употреба на графичките методи:

- Picture1.Line (x1,y1) - (x2,y2), *color*, B[F]

  На пример, црвен правоаголник целосно обоен со спротивни темиња во (2,2) и (6,4):

  Picture1.Line (2,2) - (6,4), RGB(255,0,0), BF

- Picture1.PSet (x,y), *color*

  На пример, зелена точка со координати (-3,2):

  Picture1.PSet (-3,2), RGB(0,255,0)

- Picture1.Circle (x,y), r, *color*, *start*, *end*, *aspect*

  На пример, сина елипса со центар во (0,0), a=2.5, b=5

  Picture1.Circle (0,0), 5, RGB(0,0,255), , , 1/2 или

  сино обоен кружен исечок од кужница со центар во (0,0) и радиус 5, од аголот Pi/6 до аголот Pi/3

  Picture1.Circle (0,0), 5, RGB(0,0,255), -Pi/6 , - Pi/3

  (каде Pi е однапред зададена контанта Pi=3.141592653…)

Поставување на текст

- Со задавање вредности на **својствата CurrentX и CurrentY** покажувачот се позиционира на точката со кординати (CurrentX, CurrentY) од каде започнува испишувањето на текстот со **Print методот** (позиционирањето е во горниот лев агол на текстот).

  На пример,

  ```
  Picture1.CurrentX = 13
  Picture1.CurrentY = 1
  Picture1.Print "y=0"
  ```

- Попрецизно позиционирање на текстот се врши со методите TextHeight и TextWidth.

  На пример,

  ```
  Picture1.CurrentX = 13 - Picture1.TextWidth("y=0") / 2
  Picture1.CurrentY = 1 - Picture1.TextHeight("y=0") / 2
  Picture1.Print "y=0“
  ```

  (Пример 12. Резултати од гласање.)

  (Пример 13. Цртање графици на функции (експлицитно зададени).)

Програмирање
(предавања)

Час 8. Работа со датотеки во Visual Basic (основни поими)

Кориснички дефиниран тип на податоци

- **Стрингови со фиксна должина** се декларираат со

  Dim var **As** String * n

- Со комбинирање на променливи од неколку различни типови на податоци може да се дефинира нов тип на податоци со

  [Private или Public] Type *TypeName*
  TypeElement1 As *Type1*
  TypeElement2 As *Type2*
  …
  End Type

- **Корисничкиот дефиниран тип на податоци** се користи кога сакаме да работиме со променлива која во себе чува неколку различни, но меѓусебно поврзани информации. На пример, информации за еден студент (дефинираме нов тип податоци Student).

Датотеки со случаен пристап (Random Access Files)

- **Датотеките** се состојат од записи или рекорди (Records) кои се од еден ист тип податоци, најчесто некој кориснички дефиниран тип на податоци. На пример, запис за некој студент (име, презиме, број на индекс, оценка).
- **Датотеките со случаен пристап** се како низи од записи складирани на диск. Записите се нумерирани со 1, 2, 3, и.т.н., и кон нив може да се пристапи преку нивниот реден број.
- **Отварање** на датотека со случаен пристап

  Open "filespec" For Random As #n Len = Len(record)

Датотеки со случаен пристап (Random Access Files)

- **Запишување** на податоците од променливата record на местото од br записот од датотеката отворена под број n

  Put #n, br, record

- **Читање** на податоците од br записот од датотеката отворена под број n и доделување на прочитаната вредност на променливата record

  Get #n, br, record

- Вкупниот број на карактери во датотеката отворена под број n се пресметува со **функцијата LOF(n)**, така бројот на записи во една датотека со случаен пристап е

  numberOfRecords = LOF(n) / Len(record)

- **Затварање** на датотеката

  Close #n

  (Пример 14. Резултати од испит (внесување, листање на сите, листање на положените - нова датотека).)