Ирена Стојковска

ПРОГРАМИРАЊЕ 1

1. Програмирање и програмски јазици

Компјутерот е машина која е способна да прима податоци, да ги чува и да ги обработува согласно дадените наредби. Компјутерите користат едноставен јазик, машински јазик, кој се состои од нули (0) и единици (1). Овој јазик е непристапен за корисниците, но програмата „напишана“ во овој јазик се извршува со голема брзина, затоа што тоа подразбира вклучување (=1) или исклучување (=0) прекинувачи на електрични кола.

```
01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010
01101100 01100100
```

Текстот „Hello world“ напишан на машински јазик

Програма е множество од инструкции кои сочинуваат една целина и кои го упатуваат комјутерот да извршува одредени операции за да решава поставени проблеми. Програмите се пишуваат во програмски јазик.

Програмските јазици може да бидат:
- машински јазик
- нижи (low-level) програмски јазици
- виши (high-level) програмски јазици

Нижите програмски јазици, покрај машински код, содржат и симболичен код. Тука спаѓаат: Assemler, BAL и др. Вишите програмски јазици се пофлексибилни и полесни за корстење. Користат код близок на англискиот јазик. Тука спаѓаат: Pascal, FORTRAN, COBOL, C, C++, Prolog, Java, Visual Basic и др.

Во случај на програма која не е напишана на машински јазик, специјална програма наречена компајлер ги превзема инструкциите напишани во програмскиот јазик и ги конвертира во машински јазик. Да се биде програмер, нема потреба да се знае како комјутерот преведува на машински јазик, туку треба да се знае како работи програмскиот јазик во кој се пишува програмата.

Програмскиот јазик Visual Basic е моќна алатка која најчесто програмерите ја користат за пишување, тестирање и извршување на Windows апликации, односно интерактивни модели кои претставуваат колекции од еден или повеќе фајлови комјалирани во една извршна програма. Зборот „visual“ се однесува на методот кој се користи при креирање тоа што го гледа корисникот, а „basic“ се однесува на програмскиот јазик BASIC (Beginner’s All-purpose Symbolic Instruction Code), најкористениот јазик во историјата на компјутерите. Програмата напишана во Visual Basic е VB-проект и претставува колекција од фајлови.

1.1. Основни чекори на програмирањето

При составувањето на програма која решава одреден проблем, треба да се запазат следните чекори и тоа по дадениот редослед:

1) Анализа на проблемот, одредување на влезни променливи и посакуван резултат

Во овој чекор потребно е проблемот целосно да се согледа, да се дефинираат посакуваните резултати, односно променливи кои претставуваат решение на проблемот. Треба да се разгледаат влезните променливи и како тие да се искористат при решавањето на проблемот.

2) Дефинирање на постапка за решавање на зададениот проблем

Откако ќе се согледа дека има начин како да се искористат влезните променливи за да се добие посакуваниот резултат, може да се премине на изнаоѓање на логична низа од прецизни инструкции кои го решаваат проблемот т.е. да се состави **алгоритам**. Постојат повеќе алгоритми кои решаваат еден ист проблем, и постојат и повеќе начини за прикажување на еден алгоритам. Најчесто користените се псевдо код и блог дијаграм.

Алгоритмот мора да биде што е можно подетален за да може да се согледаат сите можни ситуации или специјални случаи при решавање на проблемот. Треба да се разгледаат и случаите кога проблемот нема решение, и тие да се опфатат со алгоритмот.

3) Пишување на програма (кодирање)

Овој чекор подразбира преведување на алгоритмот во програмски јазик, за што треба да се знае синтаксата на наредбите и правилата на избраниот програмски јазик.

4) Тестирање на програмата

По кодирањето се преминува на тестирање на исправноста на програмата, односно дали таа го решава зададениот проблем. Овој чекор се однесува на логичкото тестирање, затоа што ситакстичкото тестирање се изведува при самиот процес на пишување на програмата.

При тестирањето се задаваат вредности за влезните променливи и се анализираат добиените резултати. Пожелно е уште на самиот почеток на тестирањето да се подготват соодветни влезни и излезни вредности, за да тестирањето биде поефикасно. Кај сложените програми, во оваа фаза се откриваат направените пропусти, се поправаат програмите и повторно се тестираат.

5) Документирање

Направена програма која решава даден проблем е корисна само ако е документирана. Документирањето подразбира организирање на целокупниот материјал кој ја опишува програмата: внесување коментари во кодот, опис за што служи програмата и како се користи таа, сè со цел да може да послужи на идните корисници кои не учествувале во решавањето на проблемот. Документацијата овозможува и понатамошни промени и интервенции во програмата.

Ирена Стојковска

2. Алгоритми

Алгоритам е подредена низа од прецизни и добро дефинирани инструкции, која извршува одредена задача, решава даден проблем и дава точен излез за секоја вредност на влезните променливи.

Главни карактеристики на алгоритмите се:
- алгоритмот мора да има почеток и крај,
- секој чекор на алгоритмот се извршува на единствен начин,
- опсегот на влезните променливи за кои работи алгоритмот треба да биде внимателно одреден,
- еден ист алгоритам може да биде прикажан на различни начини,
- може да постојат неколку алгоритмои 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,
- алгоритмот запира по разумно многу време.

2.1. Видови проблеми според нивното алгоритамско решение

Не секој проблем има „добро“ алгоритамско решение. Според видот на алгоритамското решение проблемите се делат на: нерешливи проблеми, „тешки“ проблеми и проблеми за кои сè уште не е најдено алгоритамско решение.

Нерешливи проблеми се оние проблеми за кои не постои алгоритам кој го решава пробчемот. Пример за таков проблем е Halting Problem или задачата на запирање која гласи: *Состави алгоритам кој одредува дали дадена програма запира по конечно многу време или работи бесконечно.*

Ќе покажеме дека таков алгоритам не постои. Да претпоставиме дека постои алгоритам A кој може да одреди дали програмата P ќе запре по конечно многу време или ќе работи бесконечно. Тогаш, $A("x := 0") = \mathsf{T}$, односно оваа програма ќе запре по конечно многу време, додека $A("\text{while } \mathsf{T} \text{ do } x := 1") = \bot$, односно оваа програма ќе работи бесконечно. Дефинираме програма $P = "\text{if } A(P) \text{ then while } \mathsf{T} \text{ do } x := 1 \text{ else } x := 0"$. Да провериме дали алгоритмот A, може да одреди дали оваа програма ќе запре по конечно многу време.

Ако A(P) = T, односно P запира по конечно многу време, па во тој случај P го извршува делот од условната структура кој е после then, т.е. $P \equiv "\text{while } \mathsf{T} \text{ do } x := 1"$, односно P ќе работи бесконечно многу време, односно A(P) = ⊥, што доведува до противречност.

Ако A(P) = ⊥, односно P работи бесконечно и во тој случај P го извршува делот од условната структура кој е после else, т.е. $P \equiv " \ x := 0 \ "$, односно P ќе сопре по конечно многу време т.е. A(P) = T, што повторно доведува до противречност.

Значи, таков алгоритам A кој може да одреди дали програмата P ќе запре по конечно многу време или ќе работи бесконечно, не постои.

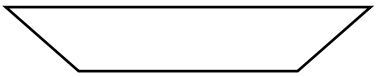
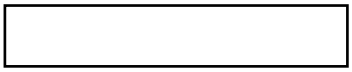
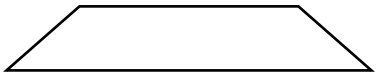
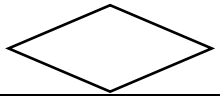
Втората класа проблеми, **„тешки“ проблеми**, се оние за кои постои алгоритамско решение, но на соодветниот алгоритам му треба многу вереме да го реши дадениот проблем. Во тој случај, најчесто се користат приближни алгоритми кои наоѓаат решенија кои приближно го решаваат проблемот. Пример за таков проблем е Traveling Salesman Problem или задачата на патувачкиот трговец која гласи: *Еден патувачки трговец треба да помине n града, секој град да го помине точно ендаш и да се врати во градот од кој тргнал. Притоа, тој треба да ги минимизира своите парни трошоци.*

Еден начин да се реши овој проблем е да се најдат сите можни тури, а потоа од нив да се најде онаа тура за која патните трошоци се минимални. Меѓутоа, за n града постојат (n-1)!/2 можни тури, што на пример за n=15 се добиваат 43 589 145 600 можни тури, и ако за една тура треба една микросекунда, тогаш ќе требаат околу 12 часа за сите тури. Оваа задача е пример за NP-задача т.е. задача со неполиномно време на решавање, што значи дека не постои начин да предвидиме колку долго би се решавала задачата, без да пробаме да решиме. Меѓутоа, постојат апроксимативни алгоритми за решавање на задачата на патувачкиот трговец, кои наоѓаат решение кое не е оптимално, но е блиску до оптималното.

Постојат и **проблеми за кои сè уште не е најдено алгоритамско решение**. Таков проблем е k-server Problem или задачата на k сервери, која гласи: *Состави онлајн алгоритам кој ќе го контролира движењето на множество од k-сервери претставени како точки во метрички простор, кои треба да ги извршат барањата, кои исто така се претставени како точки во метричкиот простор. Со пристигање на секое ново барање, алгоритмот мора да одреди кој сервер ќе се придвижикон точката на барањето. Притоа целта е да се одржи меѓусебната одалеченост на серверите што е можно помала.*

2.2. Начини на претставување на алгоритмите

Најчести начини на претставување на алгоритмите се блок дијаграм и псевдо код. **Блок дијаграмот** се состои од графички симболи поврзани со стрелки. Секој алгоритамски чекор е прикажан со соодветен графички симбол во зависност од неговото значење.

Графички симбол за алгоритамскиот чекор	**Значење на алгоритамскиот чекор**
	Поврзувачка насочена линија
	Почеток на алгоритмот
	Влезни променливи
	Процесирање/обработка
	Излезни преоменливи
	Условен алгоритамски чекор
	Крај на алгоритмот
	Поврзувач на различни насочени линии

Псевдо кодот е скратена верзија на евентуалниот комјутерски код кој му овозможува на програмерот да се фокусира на чекорите на решавање на проблемот, а не како да го користи програмскиот јазик. Предности на псевдо кодот се поголема компактност и тоа што полесно се преведува на програмскиот јазик.

2.3. Алгоритамски структури

Постојат три основни структури на алгоритмите:
- проста линиска структура,
- разгранета линиска структура,
- циклична структура.

Кај **простата линиска алгоритамска структура**, секој алгоритамски чекор се извршува точно еднаш во текот на едно извршување на алгоритмот. Со поврзување на повеќе прости линиски структури, се добива нова проста линиска структура.

Пример 1. Состави алгоритам за пресметување на растојание меѓу две точки во рамнината дадени со своите координати $A(a_1, a_2)$ и $B(b_1, b_2)$.

Кај **разгранетата линиска алгоритамска структура**, секој алгоритамски чекор се извршува најмногу еднаш. Постојат алгоритамски чекори кои не се извршуваат за време на едно извршување на програмата. Тоа се алгоритми кои содржат барем еден условен алгоритамски чекор, кој одлучува по која од гранките на алгоритмот ќе продолжи извршувањето на алгоритмот.

Пример 2. Состави алгоритам за пресметување на вредноста на y дефинирана со

$$y = y(x_1, x_2) = \begin{cases} x_1 + x_2, ако\, x_1 < x_2 \\ x_1 - x_2, ако\, x_1 \geq x_2 \end{cases}.$$

Пример 3. Состави алгоритам за пресметување на вредноста на y дефинирана со

$$y = y(x) = \sqrt{\frac{x^2 - 4}{1 - x}} + \ln(x + 5).$$

Пример 4. Состави алгоритам за пресметување на вредноста на y без користење на функција за апсолутна вредност, ако

$$y = y(x) = |1 - |x + 3| - 2x|.$$

Пример 5. Состави алгоритам за пресметување на плоштината на триаголник со дадени должини на страните a, b и c.

Основна карактеристика на **цикличната алгоритамска структура** е повеќекратно извршување на еден или повеќе алгоритамски чекори. Цилкичната алгоритамска структура содрќи условен алгоритамски чекор кој го испитува условот за крај и излез од циклусот, односно го проверува излезниот критериум. Излезниот критериум од еден циклус може да биде бројот на повторувања на циклусот или постигната точност на одреден услов.

Пример 6. Состави алгоритам за наоѓање на сите природни броеви до даден број n кои се деливи со бројот d.

Пример 7. Состави алгоритам кој ги наоѓа сите Фибоначиеви броеви до даден број n.

Пример 8. Состави алгоритам за пресметување на аритметичката средина на дадени n броеви $a_1, a_2, ..., a_n$.

Сложена алгоритамска структура се добива со замена на основните алгоритамски чекори во елементарните алгоритамски структури со други елементарни структури. Сложеноста се зголемува кога се користат повеќе од еден циклус, еден по друг или вгнездени еден во друг.

Пример 9. Состави алгоритам за наоѓање на сите прости броеви до даден број n.

Пример 10. Состави алгоритам за пресметување на вредноста на сумата

$$S = S(n) = \frac{1}{1!} + \frac{1+2}{2!} + \frac{1+2+3}{3!} + ... + + \frac{1+2+...+n}{n!}.$$

2.4. Задачи

1. Состави алгоритам за пресметување на вредноста на z дефинирана со

$$z = z(x,y) = \begin{cases} \min\{-|x|, y\} & , y < 0 \\ -|x| & , y = 0 \\ \max\{-|x|, -y\} & , y > 0 \end{cases}.$$

2. Состави алгоритам за решавање во $\mathbb{R}$ на квадратната равенка $ax^2 + bx + c = 0$, каде $a, b, c \in \mathbb{R}$ се такви да $a \neq 0$.

3. Состави алгоритам за пресметување на збирот на првите n непарни природни броеви.

4. Состави алгоритам за пресметување на вредноста на сумата

$$S = S(n) = 1^2 - 2^2 + 3^2 - 4^2 + \ldots + (2n-1)^2 - (2n)^2 \ .$$

5. Состави алгоритам за нашѓање на сите решенија во $\mathbb{N}^2$ на Диофантовата равенка $ax + by = c$, за дадени $a, b, c \in \mathbb{N}$. На пример, за $a = 2, b = 1, c = 8$, сите решенија на $2x + y = 8$ во $\mathbb{N}^2$ се: (1, 6), (2, 4) и (3, 2).

6. Состави алгоритам кој за даден почетен природен број a_0 испишува низа броеви добиена според правилото

$$a_{n+1} = \begin{cases} a_n / 2 & , a_n - \text{парен} \\ 3a_n + 1 & , a_n - \text{непарен} \end{cases}, \ n = 0, 1, 2, \ldots$$

сè додека не ја добие низата 4, 2, 1 или сè додека не го надмине дозволениот максимален број членови во низата $n_{\max}$.

Ирена Стојковска

3. Основи на програмирањето во Visual Basic

Податок е запишана информација која комјутерот ја обработува. Постојат два основни типа на податоци: нумерички и текстуални податоци. **Нумеричките податоци** претставуваат нумеричка вредност изразена во декаден броен систем, а **текстуалните податоци** претставуваат низа од симболи сместена меѓу знаци за наводници.

Податоците се складираат во константи и променливи. **Константите** имаат постојана вредност за цело време на извршување на програмата, примери за константи се бровите, а со наредбата **Const** *constname = expression* може да се дефинира нова константа.

Променлива е симболична ознака на која може да ѝ се додели вредност од некој податок, т.е со помош на променливата пристапуваме кон содржината на податокот. Во зависност од природата на податоците, постојат нумерички и текстуални променливи. Основни карактеристики на секоја променлива се: име на променливата и моментална вредност на променливата. **Името на променливата** може да биде составено најмалку од 1 и најмногу од 255 карактери (симболи), мора да започне со буква, може да содржи букви (мали или големи), цифри и подвлечени линии ("_"), притоа Visual Basic не прави разлика меѓу малите и големите букви во името на променливата (имињата brojnacifri и BrojNaCifri се однесуваат на една иста променлива). Доделувањето **вредност на променлива** се изведува со симболот "=“. Притоа со:

1) var = num, на променливата var ѝ се доделува нумеричката вредност num
   пр. a = 3
2) var = str, на променливата var ѝ се доделува текстуалната вредност str
   пр. ime = ”Ana”
3) var2 = var1, на променливата var2 ѝ 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

Во примерите за третата и четвртата потточка, искористен е симболот “:” кој означува разделување на два програмски реда, ако тие се запишани еден до друг.

Декларирањето на променливи се изведува со **Dim** *varname* **As** *type*, каде *varname* е името на променливата, а *type* е типот на променливата. Декларирањето се извршува на почеток од процедурата, или во делот General Declarations за да важи за сите процедури од формата, или во модул (посебен тип на фајл во состав на VB-проектот) за да важи за сите форми во состав на VB-проектот. Декларирањето на променливи 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.

Во Visual Basic може да се декларираат следните **типови на променливи** (дадени со опсегот на вредности кои може да се складираат во нив):

- Integer - цели броеви од -32768 до 32767
- Long - цели броеви од -2147483648 до 2147483647
- Single - бројот 0, броевите од 1.40129·10-45 до 3.40283·1038 со најмногу 7 значајни цифри и негативите на овие броеви
- Double - бројот 0, броевите од 4.9406520·10-324 до 1.797693134862316·10308 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

Добивањето на нов податок од неколку дадени податоци се нарекува **израз**. Изразот може да биде нумерички или текстуален, а начинот на кој од дадените податоци се добива нов податок е со помош на **операциите за нумерички и текстуални податоци**:

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

Во изразите со текстуални податоци, дозволено е да се користи и операцијата „+“ како операција за слепување.

Ако изразот содржи повеќе од една операција, тогаш вредноста на изразот се одредува така што операциите се извршуваат од лево на десно, водејќи сметка за **приоритот на операциите**:

Приоритет на извршување на операциите:	
()	Вредноста во загради прва се пресметува
^	Дигањето на степен е второ по приоритет
-	Создавањето негативен број е трето
* /	Множењето и делењето се четврти и се извршуваат по редослед на појавување
\	Целобројното делење е петто
Mod	Остатокот при делење е шести
+ -	Собирањето и одземањето се последни и се извршуваат по редослед на појавување

Во рамки на еден израз може да се користат и **операции за подредување** и во тој случај изразот добива вредност True (точно) или False (неточно), односно се добива податок од типот Boolean.

Операции за подредување:	
=	енакво на
<>	различно од
<	помало од
>	поглемо од
<=	помало или еднакво на
>=	поголемо или еднакво на

Во изразите може да се користат и **логичките опрации**:

Логичкин операции:	
Not	не (негација)
And	и (конјункција)
Or	или (дисјункција)
Xor	или...или (исклучна дисјункција)
Eqv	ако и само ако (еквиваленција)
Imp	ако...тогаш (импликација)

Visual Basic има повеќе предефинирани функции. 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 текстуални. **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.

3.1. Наредби за влез/излез

Функцијата **InputBox** овозможува да се направи влез на текстуални или нумерички вредности и тие да се доделат на одредена променлива. При повикување на оваа функција се отвара комуникациски прозорец преку кој се внесува вредноста. Основниот облик на синтаксата на оваа функција е:

1) var = **InputBox**(*prompt*) за доделување на тесктуална вредност на променивата var

2) var = **Val**(**InputBox**(*prompt*)) за доделување на нумеричка вредост на променивата var

Притоа *prompt* е текст кој излегува во комуникацискиот прозорец.

Излезот, односно испишувањето на вредностите на податоците на формата или друг објект се извршува со наредбата **Print**. Основната синтакса на оваа наредба е:

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

Доколу вредностите (*outputlist*) се испишуваат на формата, не се наведува ништо на местото од *object*. Доколку се изостави *outputlist* излезот е празен ред. Излезот може да се уреди со функциите Spc(n) и Tab(n) и сибмолите , (запирка) и ; (точка запирка).

За излез може да се користи и функцијата **MsgBox** со која се отвара комуникациски прозорец во кој се прикажува излезот. Оваа функција се користи за да се привлече вниманието на корисникот. Нејзината основна синтакса е:

MsgBox(*prompt*)

Притоа *prompt* е текст кој излегува во комуникацискиот прозорец, односно излезот.

Пример 1. Побарај од корисникот да внесе име и презиме, а потоа излезот прикажи го како „Вашето име и презиме се:...“.

Пример 2. Побарај од корисникот да ја внесе годината на раѓање, а потоа излезот прикажи го како „Оваа година (ќе) имате .... години.“.

Пример 3. За внесени природни броеви *a* и *b* најди ги количникот *k* и остатокот *r* при делење на *a* со *b*. Потоа, излезот прикажи го како $a = k \cdot b + r$.

Пример 4. За внесен трицифрен број одреди ги цифрите на местото на стотки (С), десетки (Д) и единици (Е).

3.2. Условна структура If…Then

Условната структура If…Then овозможува гранање на програмата во зависност од условот. **Прост облик** на условната структура If…Then е

If *cond* **Then** *statement1* [**Else** *statement2*]

Кога при извршувањето, програмата наиде на оваа наредба, прво се проверува дали важи условот *cond*, ако важи тогаш се извршува наредбата *statement1*, а ако не важи се извршува наредбата *statement2*. Притоа, делот Else не е задолжителен. Доколку не е исполент условот *cond* и не постои делот Else, тогаш програмата продолжува да ги извршува следните редови.

Сложениот облик на условната структура If…Then е

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

При извршување на сложената уловна структура If…Then, прво се проверува точноста на условот *cond1*, ако е исполент се извршуваат наредбите *statements1* и програмата продолжува да ги извршува наредбите после End If. Ако не е исполнет условот *cond1*, се проверува дали е исполнет условот *cond2*, ако е исполент се извршуваат наредбите *statements2* и програмата продолжува да ги извршува наредбите после End If итн. Ако не се исполнети никој од условите *cond1, cond2, …, cond(n)*, тогаш ако постои делот Else, се извршуваат наредбите *statements(n+1)*.

Треба да се внимава да се избегнуваат вгнездени условни структури во случај кога условите се независни, односно кога може да им се провери нивната точност независно еден од друг. Така, условната структура

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

може да се презашише како

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

Пример 5. За дадени вредности на x_1, x_2 , пресметај ја вредноста на

$$y = \begin{cases} x_1 + x_2 & , x_1 < x_2 \\ x_1 \cdot x_2 & , x_1 = x_2 \\ x_1 / x_2 & , x_1 > x_2, x_2 \neq 0 \end{cases}$$

Пример 6. (со интерфејс) Состави програма за погодување „замислен“ број од 1 до 1000.

3.2.1. Задачи за вежби

Задача 1. За дадена вредност на x пресметај ја вредноста на

$$y = y(x) = \sqrt{\frac{x-2}{x+1}} - \log_{10}(x+5) + \cos^2(x+2).$$

Задача 2. За дадена вредност на x пресметај ја вредноста на

$$y = y(x) = \sqrt[3]{\frac{x+5}{x-2}}.$$

Задача 3. За дадени вредности на x и y пресметај ја вредноста на

$$A = A(x,y) = \begin{cases} \min\{x-y, x+y\} & , x \le y \\ \max\{x-y, x+y\} & , x > y \end{cases}$$

Задача 4. Најди го најмалиот од броевите a, b и c.

Задача 5. Подреди ги по големина броевите a, b и c, од најмалиот кон најголемиот.

3.3. Условна структура Select Case

Еден начин на избегнување на вгнездени условни структури If…Then и добивање поголема прегледност на програмата е со условната структура Select Case со синтакса:

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

Притоа, изборите зависат од селекторот (selector) кој може да биде нумерички или текстуален израз. Листата вредности (varlist) која го одредува изборот може да содржи една или повеќе вредности оделени со запирка, како и клучните зборови **To** и **Is** (на пример, 3,4,5 или 3 To 5 или Is <=5).

Пример 7. „Прочитај“ го внесениот двоцифрен број n (на пример, n=65 прочитај го како „шест пет“).

Пример 8. Креирај едноставен калкулатор кој за два внесени броја a и b ќе пресмета нивен збир, разлика, производ или количник.

Пример 9. Одреди дали внесена буква е самогласка или согласка.

Пример 10. Во зависност од освоените поени на испит, одреди ја оценката за студентот (0-49 оценка 5, 50-59 оценка 6, 60-69 оценка 7, 70-79 оценка 8, 80-89 оценка 9, 90-100 оценка 10).

Пример 11. Ако се дадени денот, месецот и годината на раѓање на неке лице, најди го неговиот примарниот број на раѓање. (На пример, нека некое лице е родено на 23.04.2002, тогаш прво ги собираме сите цифри 2+3+0+4+2+0+0+2=13, потоа на добиениот збир му ги собираме цифрите сè додека не добиеме едноцифрен број, па бидејќи 1+3 = 4, па примарниот број на раѓање е 4.) Потоа, „одреди го карактерот“ на лицето според неговиот примарен број на раѓање.

3.4. Задачи за самостојна рабита

Задача 6. За дадени вредности на x пресметај ја вредноста на

а) $y = y(x) = \sqrt{\frac{2x^2+x}{\frac{x^2+1}{x}-3}}$; б) $y = y(x) = \frac{\sqrt{x+5}}{x^2-4} - 2x + \ln(\sqrt{3-x})$.

Задача 7. За дадени вредности на x и y, пресметај ја вредноста на изразот

а) $A = A(x,y) = e^{-\frac{x^2+3}{2-y}} - \sqrt{\frac{2-y^2}{x+4}}$; б) $A = A(x,y) = \frac{\min\{x,y\}+0.5}{1+\max^2\{x,y\}}$

Задача 8. Подреди ги по големина броевите a, b, c и d, од најмалиот кон најголемиот.

Задача 9. Во зависност од годината на раѓање, одреди дали лицето е во градинка (0-5 години), основно училиште (6-14 години), средно училиште (15-18 години), студент (19-22 години), работник (23-62 години) или пензионер (>62 години).

Задача 10. Во зависност од редниот број на месецот (1-12) одреди го годишното време (3-5 пролет, 6-8 лето, 9-11 есен, 12,1,2 зима).

Задача 11. Реши ја во $\mathbb{R}$ квадратната неравенка $ax^2+bx+c\,\rho\,0$, каде $\rho \in \{<,>\}$ и $a \neq 0$. Помош: Влезни вредности се броевите a,b,c и знакот ρ на неравенката, а излез е интервалот од решенија на дадената неравенка.

Задача 12. За интервалите $(a,b),(c,d) \subseteq \mathbb{R}^2$, најди ги нивната унија, пресек и разлика.

Задача 13. Реши го во $\mathbb{R}^2$ системот од линеарни равенки $\begin{cases} a_1x+b_1y=c_1 \\ a_2x+b_2y=c_2 \end{cases}$ со помош на Крамеровите формули.
Помош: Влезни вредности се броевите a_1,b_1,c_1,a_2,b_2,c_2, а излез е решението (x,y) или соопштение дека системот нема решение или дека има бесконечно многу решенија. Во случај на бесконечно многу решенија да се даде обликот на решенијата.

Задача 14. За внесен датум, одреди го денот во неделата, според алгоритмот опишан на следниот линк https://matematika-plus.weebly.com/kako-da-presmetate-vo-koj-den-od-nedelata-ste-rodeni.html.

3.5. Циклична структура For…Next

Цикличната структура For…Next се користи кога се знае точно колку пати треба да се изврши една група наредби. Синтаксата е:

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

каде *counter* (бројач) е некоја нумеричка променлива, *start* (почеток) е почетна вредност, а *end* (крај) е крајната вредност на бројачот, *step* (чекор) е показател за колку бројачот се менува при секое повторување на циклусот (вредноста на чекорот може да е и негативна).

Треба да се избегнува промена на вредноста на бројаќот внатре во циклусот, бидејќи може да се креира бескраен циклус. Дозволено е вгнездување на друг For…Next циклус, но не смее да има преклопување и бројачот треба да е различен. Со Exit For привремено може да се излезе од For циклусот.

3.6. Циклична структура Do…Loop

Со цикличната структура Do…Loop се повторуваат група наредби сè додека е точен некој услов и сè додека не стане точен некој услов. Постојат неколку синтакси на оваа циклична структура:

1) **Do While** *cond*
  statements
Loop

2) **Do**
  statements
Loop While *cond*

3) **Do Until** *cond*
  statements
Loop

4) **Do**
  statements
Loop Until *cond*

каде *cond* (услов) е условот кој се проверува дали е исполнет и се извршува блокот наредби (statements), ако е исполнет - случаите 1) и 2), или ако не е исполнет – случаите 3) и 4). Разликата меѓу 1) и 2) е што во 2) блокот наредби (statements) се извршува еднаш, а потоа се проверува дали е исполнет условот (*cond*). Слично и за разликата меѓу 3) и 4).

Дозволено е вгнездување на друг Do…Loop циклус, а предвремен излез од циклусот може да се направи со Exit Do. Доколку при извршување на програмата се случи бескраен циклус, тој може да се прекине со Ctrl+Break или Break копчето од лентата со инструменти.

Пример 12. За внесен број n отпечати ги сите природни броеви заклучно до n.

Пример 13. За внесен број n отпечати ги прво сите парни броеви, а потоа сите непарни броеви не поголеми од n.

Пример 14. Отпечати ги сите броеви меѓу m и n кои не се деливи ниту со d_1, ниту со d_2.

Пример 15. Најди го збирот на првите n природни броеви т.е.

$$S_n = 1+2+3+\ldots+n = \sum_{i=1}^{n} i \ .$$

Пример 16. Најди го збирот на првите n непарни природни броеви т.е.

$$S_n = 1+3+5+\ldots+(2n-1) = \sum_{i=1}^{n} (2i-1) \ .$$

Пример 17. За даден број n пресметај $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$.

Пример 18. За даден број n пресметај $n!! = \begin{cases} 2 \cdot 4 \cdot 6 \cdot \ldots \cdot n & n - \text{парен} \\ 1 \cdot 3 \cdot 5 \cdot \ldots \cdot n & n - \text{непарен} \end{cases}$.

Пример 19. Одреди го бројот на цифри на бројот n.

Пример 20. Најди го збирот на цифри на бројот n.

Пример 21. Најди го обратниот број на бројот n. Дали n е палиндром? (Палиндром е број кој е еднаков на неговиот обратен број т.е. бројот запишан со истите цифри, но во обратен редослед, на пример бројот 14541 е палиндром.)

Пример 22. Најди го бројот на различни цифри од кои е составен бројот n. Кои се тие цифри?

3.7. Задачи за самостојна работа

Задача 15. Отпечати ги квадратите на сите природни броеви заклучно до n.
(ТЕСТ: n=10 треба да даде излез: 1, 4, 9, 14, 25, 36, 49, 64, 81, 100)

Задача 16. Најди го збирот на броевите меѓу m и n кои се деливи со d.
(ТЕСТ: m=15, n=100, d=10 треба да даде излез: 540)

Задача 17. За дадени n и k пресметај ја вредноста на биномниот коефициент

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} .$$

(ТЕСТ: n=6, k=4 треба за даде излез: 15)

Задача 18. Запиши го бројот n во развиена форма.
(ТЕСТ: n=1643 треба да даден излез: 1643 = 1000 + 600 + 40 + 3, додека n=805 треба да даде излез 805 = 800 + 5)

Задача 19. Најди го збирот на цифри на бројот n кои се наоѓаат на непарните места сметано од лево на десно.
(ТЕСТ: n=186543 треба да даде излез: 11)

Задача 20. Провери дали два внесени броја n и m се анаграми еден на друг. (Анаграми се броеви кои се составени од истите цифри, само може да се по друг редослед.)
(ТЕСТ: n=2642, m= 4262 дава излез: ДА, додека n=8938, m=8393 дава излез: НЕ)

3.7. Задачи со If…Then, Select…Case, For…Next и Do…Loop

Задача 21. Најди најмал заеднички содржател (НЗС) и најголем заеднички делител (НЗД) на броевите a и b.

Задача 22. Со помош на Евклидовиот алгоритам најди НЗД на броевите a и b. Евклидов алгоритам: За a≥b важи НЗД(a,b) = НЗД(b,r), каде r е остатокот при делење на a со b, т.е. a = b * k + r, 0≤r≤b-1. Нови вредности за a и b се земаат b и r соодветно и постапката се повторува сè додека b не стане делител на a, и тогаш бараниот НЗД е b. На пример, НЗД(156, 18) = НЗД(18, 12) = НЗД(12, 6) = 6.

Задача 23. Најди го збирот на дропките a/b и c/d (a, b, c, d се цели броеви т.ш. b, d ≠0) и доведи го до нескратлива дропка.

Задача 24. Најди ги сите совршени броеви не поголеми од бројот n. (Еден број е совршен, ако ееднаков на збирот на сите свои вистински делители, тоа се делители на бројот различни од самиот тој број. На пример, 6 = 1 + 2 + 3, 28 = 1 + 2 + 4 + 7 + 14.)

Задача 25. Најди ги сите трицифрени броеви кои се еднакви на збирот од кубовите на своите цифри (Амстронгови броеви).

Задача 26. Најди ги сите n-цифрени броеви кои се деливи со d и во својот декаден запис не ја содржат цифрата c.

Задача 27. Најди ги сите Питагорови тројки брови не поголеми од n. (Тројката (a, b, c) е Питагорова тројка броеви, ако важи $a^2 + b^2 = c^2$.)

Задача 28. Најди ги сите парови заемно прости броеви не поголеми од n. (Броевите a и b се заемно прости, ако НЗД(a, b)=1.)

Задача 29. Пресметај ја приближната вредност на π сметајќи: 1) до k -тиот член; 2) со точност $\varepsilon > 0$, со помош на развојот на функцијата

а) $\operatorname{arctg} x$ б) $\arcsin x$

Упатство.

а) $\operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - ... + (-1)^k \frac{x^{2k+1}}{2k+1} + ...$

$x = 1 \Rightarrow \pi = 4 \cdot (1 - \frac{1}{3} + \frac{1}{5} - ... + (-1)^k \frac{1}{2k+1} + ...)$

За барањето под 2), бидејќи редот е алтернативен, сумирањето се изведува сè додека член кој по апсолутна вредност е поголем или еднаков на точноста $\varepsilon > 0$.

б) $\arcsin x = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2^2 \cdot 2} \cdot \frac{x^5}{5} + ... + \frac{1 \cdot 3 \cdot ... \cdot (2k-1)}{2^k \cdot k!} \cdot \frac{x^{2k+1}}{2k+1} + ...$

$x = \frac{1}{2} \Rightarrow \pi = 6 \cdot (\frac{1}{2} + \frac{1}{2^4 \cdot 3} + \frac{1 \cdot 3}{2^7 \cdot 2 \cdot 5} + ... + \frac{1 \cdot 3 \cdot ... \cdot (2k-1)}{2^{3k+1} \cdot k! \cdot (2k+1)} + ...)$

За барањето под 2), сумирањето се изведува сè додека остатокот по апсолутна вредност не стане помал од точноста $\varepsilon > 0$. Се покажува дека $|R_k| < \frac{2}{2k+3}$, па сумирањето се изведува сè додека $\frac{2}{2k+3} \geq \varepsilon$.

Задача 30. За дадено n пресметај го производот

$$\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2+\sqrt{2}}}\cdot\frac{1}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\cdot\ldots\cdot\underbrace{\frac{1}{\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}}}_{n\ \text{корени}}\cdot$$

3.8. Задачи за самостојна работа

Задача 31. Најди најмал заеднички содржател (НЗС) и најголем заеднички делител (НЗД) на броевите a, b и c.

Задача 32. Најди ги сите парови пријателски броеви не поголеми од n. (Два броја се пријателски ако едниот број е еднаков на збирот од вистинските делители на другиот број и обратно. На пример, броевите 220 и 284 се пријателски броеви затоа што 220 = 1 + 2 + 4 + 71 + 142 т.е. збирот на вистинските делители на 284, но и 284 = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 т.е. збирот на вистинските делители на 220.)

Задача 33. Најди ги сите n-цифрени броеви кои имаат еднакви прва и последна цифра и чиј збир на цифри е еднаков на s.

Задача 34. Најди ги сите парови од прости броеви близнаци не поголеми од n. (Два прости броја кои се разликуваат за 2 се прости броеви близнаци. Првите парови прости броеви близнаци се: (3, 5), (5, 7), (11, 13), (17, 19) итн.)

Задача 35. Пресметај ја приближната вредност на $\ln 2$ сметајќи: 1) до k -тиот член; 2) со точност $\varepsilon > 0$, со помош на развојот на функцијата

а) $\ln(1+x)$ б) $\ln(\frac{1-x}{1+x})$

Упатство.

а) $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{k+1}\frac{x^k}{k} + \ldots$

б) $\ln(\frac{1-x}{1+x}) = -2\cdot(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots + \frac{x^{2k+1}}{2k+1} + \ldots)$

Задача 36. За дадено n, пресметај ги сумите:

а) $S_n = \frac{2}{1\cdot 3} + \frac{3}{2\cdot 4} + \frac{4}{3\cdot 5} + \ldots + \frac{n+1}{n(n+2)}$

б) $S_n = \frac{1}{1\cdot 2\cdot 3} - \frac{1}{3\cdot 4\cdot 5} + \frac{1}{5\cdot 6\cdot 7} - \ldots + (-1)^{n+1}\cdot\frac{1}{(2n-1)\cdot 2n\cdot(2n+1)}$

Задача 37. За дадени n и x, пресметај ги сумите:

а) $S_n = \sin(x) + \sin^2(x) + \ldots + \sin^n(x)$

б) $S_n = \cos(x) + \cos(x^2) + \ldots + \cos(x^n)$

Задача 38. За $x \in \mathbb{R}$, приближно пресметај ја сумата $S = x - \frac{x^3}{3!} + \ldots + (-1)^n\frac{x^{2n+1}}{(2n+1)!} + \ldots$, со точност $\varepsilon > 0$ (сумирањето е сè додека $|(-1)^n\frac{x^{2n+1}}{(2n+1)!}| \geq \varepsilon$).

Ирена Стојковска

4. Работа со текстуални податоци (Strings)

Текстуален податок е низа од симболи која се третира како една целина. При работа со конкретни текстуални податоци таа низа од симболи ја сместуваме меѓу наводници, на пример “a”, “B12”, “avtomobil”, “12345” и “Denes e ubav den.” се примери за текстуални податоци. Празното место е исто така симбол, а и наводниците претставуваат симбол, кој може да биде дел од текстуален податок, а како, ќе видиме подоцна. **Нулти стринг** или **празен стринг** е текстуалниот податок кој не содржи симболи (“”).

Типот на променливи резервиран за текстуалните податоци е **String** со кој се складираат текстуални податоци од 0 до 32767 симболи. При декларирањето на променливите од типот String може да се додели и фиксна должина (вкупен број на симболи) на текстуалните податоци кои се складираат во таа променлива. На пример, со програмските редови

```
Dim a As String
Dim b As String*10
a = “Denes e ubav den.”
b = “Denes e ubav den.”
```

во променливата a ќе се складира текстуалниот податок “Denes e ubav den.”, а во променливата b само првите 10 симболи, односно “Denes e ub”, сметајќи го и празното место за симбол.

Операцијата за работа со текстуални податоци и променливи е операцијата за спојување, односно слепување (&). **Текстуален израз** е комбинација од текстуални податоци, текстуални променливи и знаци за спојување (&) која дава резултат една текстуална вредност. На пример, во програмските редови

```
a = “abc”
b = a & “123” & a
```

текстуален израз е: a & “123” & a, а променливата b по извршување на двата програмски реда ќе добие вредност “abc123abc”.

4.1. ASCII карактерно множество (0-255)

Секој симбол има свој ASCII (American Standard Code for Information Interchange) код. ASCII карактерното множество ги содржи сите симболи подредени според нивните ASCII кодови. Така на пример, цифрите 0-9 има ASCII кодови 48-57 соодветно, големите латинични букви A-Z имаат ASCII кодови 65-90 соодветно, малите латинични букви a-z имаат ASCII кодови 97-122 соодветно, празното место има ASCII код 32, а наводниците имаат ASCII код 34.

Фукцијата **Asc** го дава ASCII кодот на симболот, а функцијата **Chr** го дава симболот кој одговара на дадениот ASCII код.

Пример 1. Прикажи ги симболите кои одговараат на ASCII кодовите 32-126. Приказот да биде со ASCII кодот и соодветниот симбол, по неколку такви парови во еден ред.

Бидејќи ASCII кодот на наводници е 34, ако сакаме во некоја променлива да складираме наводници или да отпечатиме наводници, тогаш ќе треба да се послужиме со функцијата Chr, односно има разлика во излезот на следните програмски редови

```
Print “OOU ” & “Dimitar Miladinov” & “, Skopje”
Print “OOU ” & Chr(34) & “Dimitar Miladinov” & Chr(34) & “, Skopje”
```

кои соодветно даваат излез

```
OOU Dimitar Miladinov, Skopje
OOU "Dimitar Miladinov", Skopje
```

Примената на ASCII кодовите слиужи и за споредба на текстуални податоци. Операциите за подредување =, <>, <, >, <= и >= се применуваат така што редоследно се споредуваат ASCII кодовите почнувајќи од првиот. На пример, “ABC” < ”abc” затоа што Asc(“A”) < Asc(“a”), потоа “Andrej” > ”Andrea” затоа што првите пет симболи се исти, но Asc(“j”) > Asc(“a”).

Пример 2. Внеси име и презиме за три студенти, а потоа подреди ги според презимето по азбучен редослед (A-Z).

4.2. Функции за работа со текстуални податоци

Во следната табела се дадени основните функции за работа со стрингови.

Функција	Опис	Пример
Left (string, n)	Од string издвојува n симболи од лево.	Left (“abcde”, 3) излез: “abc”
Right (string, n)	Од string издвојува n симболи од десно.	Right (“abcde”, 3) излез: “cde”
Mid (string, i, n)	Почнувајќи од i-тиот симбол од string издвојува n симболи.	Mid (“abcde”, 3, 1) излез: “c”
UCase (string)	Ги презапишува сите букви во големи.	UCase (“Visual Basic 6.0”) излез: “VISUAL BASIC 6.0”
LCase (string)	Ги презапишува сите букви во мали.	UCase (“Visual Basic 6.0”) излез: “visual basic 6.0”
Trim (string)	Ги отстранува сите празни места од лево и од десно.	Trim (“ aa bb cc “) излез: “aa bb cc”
LTrim (string)	Ги отстранува сите празни места од лево.	Trim (“ aa bb cc “) излез: “aa bb cc ”
RTrim (string)	Ги отстранува сите празни места од десно.	Trim (“ aa bb cc “) излез: “ aa bb cc”
String (n, character)	Го запишува n пати симболот character.	String (10, “*”) излез: “**********”
Str (number)	Го конвертира нумеричкиот податок number во тесктуален податок.	Str (12345) излез: “12345”

Len (string)	Ја дава должината т.е. бројот на симболи во string.	Len (“Visual Basic 6.0”) излез: 16
InStr ([start,] string1, string2)	Почнувајќи од start позицијата во string1, бара од која позиција се сречаќава string2 како подстринг.	InStr (1, “banana”, “ana”) излез: 2 InStr (3, “banana”, “ana”) излез: 4 InStr (5, “banana”, “ana”) излез: 0

Пример 3. Одреди колку пати дадена буква се среќава во даден збор.

4.3. Задачи

Задача 1. Одреди колку збора има во една реченица.

Задача 2. Отстрани ги сите празни места во дадена реченица.

Задача 3. Провери дали даден збор е палиндром т.е. дали се чита исто и од напред и од назад. На пример, зборот “anavolimilovana” е палиндром, но зборот “ananas” не е палиндром.

Задача 4. Одреди го бројот на различни букви во даден збор.

Задача 5. Провери дали два збора се анаграми еден на друг. Анаграми се зборови кои се составени од истите букви, но по различен редослед. На пример, “marka” и “karma” се анаграми, но “marka” и “rakun” не се анаграми.

Задача 6. Најди ја „разликата“ на два збора со еднаква должина т.е. бројот на букви по кои се разликуваат. На пример, разликата на зборовите “ucenici” и “uciteli” е 4, бидејќи им се разликуваат буквите на трето, четврто, петто и шесто место.

Задача 7. За два внесени збора најди го заедничкиот подстринг со најголема должина. На пример, за зборовите “ananas” и “banana” заеднички подстринг со најголема должина е “anana”.

Задача 8. Кодирај даден текст според Цезарова шифра со чекор k, што значи дека секоја буква се презапишува со буквата која во азбуката е k места после неа, а кога ќе се стигне до z со презапишвањето, азбуката почнува од почеток, односно од a. На пример, ако чекорот е k=4, тогаш a->e, b->f, c->g, …, v->z, w->a, x->b, y->c и z->d, па реченицата „Denes e ubav den.” ќе се презапише во “Hiriw i yfez hir.”

4.4. Задачи за самостојна работа

Задача 9. Одреди кој од два збора е "позвучен", односно кој има поголем број на самогласки во однос на неговата должина. (на пример, зборот "каша" е позвучен од зборот "брсјаците", затоа што првиот збор има 2 самогласки од вкупно 4 букви, а вториот има 3 самогласки од вкупно 9 букви, и притоа 2/4 > 3/9).

Задача 10. Најди ги сите букви во еден збор кои се наоѓаат меѓу две самогласки. Колку такви букви има? (На пример, во зборот "самогласка", таква е само буквата "м", додека во аборот "камата", такви се буквите "м" и "т".)

Задача 11. Напиши програма која „чита“ цифри т.е. цифрите 0 до 9 го заменува со зборови „nula“ до „devet“. На пример, реченицата „Martin ima 2 kutii, a vo sekoja od kutiite ima po 15 dzamlii.“, да се замени со реченицата „ Martin ima dva kutii, a vo sekoja od kutiite ima po edenpet dzamlii. “

Задача 12. Презапиши даден текст на „па-пе-пи-по-пу“ јазик, што значи да после секоја самогласка се додаде соодветниот слог кој завршува на таа самогласка. На пример, реченицата „Denes e ubav den.” презапишана во тој јазик е “Depenepes epe upubapav depen.”

Задача 13. Во дадена реченица замени ги повеќекратните празни места со едно празно место. Потоа, издвои ги зборовите и запиши ги еден по друг.

Задача 14. Презапиши даден текст, така што на секој збор ќе му ги замениш правата и последната буква. На пример, реченицата “Denes e ubav soncev den.” презапишана на тој начин ќе биде “Sened e vbau vonces ned.”

Ирена Стојковска

5. Низи од променливи (Arrays)

Променлива (проста променлива) е симболична ознака на која може да ѝ се додели некоја вредност. **Низа од променливи** е колекција од прости променливи од ист тип на која ѝ се доделува листа од вредности. При декларирање на една низа од променливи треба да се зададат: името на низата, типот на променливите, димезијата на низата и бројот на променливи по секоја димензија. Со низата од променливи може да се работи само ако е целосно декларирана. Во зависност од начинот на декларирање на низата, разликуваме **низи со фиксна голема** и **динамички низи**.

Кај **низите со фиксна големина** сите компоненти се декларираат одеднаш. Еднодимензионални низи се декларираат на следниот начин:

Dim *arrayName* (n) **As** *Type*
Dim *arrayName* (m To n) **As** *Type*

каде *arrayName* е името на низата, *Type* е типот на променливите, при тоа првата низа содржи n променливи: *arrayName* (0), *arrayName* (1), ..., *arrayName* (n-1), а втората низа ги содржи променивите: *arrayName* (m), *arrayName* (m+1), ..., *arrayName* (n), за $m \leq n$. При декларирање на дводимензионални (и повеќедимензионални) низи, треба да се зададе и бројот на променливи по секоја од димензиите. На пример,

Dim A (2, 3) As Integer

содржи $2 \cdot 3 = 6$ променливи од типот Integer, и тоа: A(0, 0), A(0, 1), A(0,2), A(1, 0), A(1, 1), A(1,2), додека пак низата

Dim B (1 To 3, 1 To 2) As Double

ги содржи променливите: B(1, 1), B(1, 2), B(2, 1), B(2, 2), (3, 1), B(3, 2).

Пример 1. Пресметај го просекот од положените предмети од првиот семестар на еден студент од прва година.

Пример 2. Отпечати матрица на меѓусебни растојанија меѓу четири града.

Низите од променливи со променлива големина, односно **динамичките низи**, на почетокот се декларираат само со името на низата и типот на променливите:

Dim *arrayName* () **As** *Type*

каде *arrayName* е името на низата, а *Type* е типот на променливите. Потоа, се додекларираат (ReDim) и димензијата и бројот на променливи по секоја од димензиите:

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

Доколку сакаме да ги задржиме веќе доделените вредности, тогаш се користи додекларирање со задршка (ReDim Preserve), односно:

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

Пример 3. За дадени n позитивни реални броеви $a_1, a_2, ..., a_n$, пресметај ги аритметичката A_n, геометриската G_n, хармониската H_n и квадратната средина K_n т.е.

$$A_n = \frac{a_1 + a_2 + ... + a_n}{n},\ G_n = \sqrt[n]{a_1 \cdot a_2 \cdot ... \cdot a_n},\ H_n = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + ... + \frac{1}{a_n}},\ K_n = \sqrt{\frac{{a_1}^2 + {a_2}^2 + ... + {a_n}^2}{n}}.$$

5.1. Задачи со еднодимезионални низи од променливи

Задача 1. Најди ги најголемиот и најмалиот елемент на дадена низа броеви $a_1, a_2, ..., a_n$.

Задача 2. Подреди ја низата броеви $a_1, a_2, ..., a_n$ од најмалиот до најголемиот број со помош на алгоритмот Selection Sort.

Задача 3. Одреди го бројот на промени на знак на низата $a_1, a_2, ..., a_n$.

Задача 4. Одреди го членот од низата $a_1, a_2, ..., a_n$ кој е најблизок до дадена вредност b.

Задача 5. Најди ги сите прости броеви до бројот n со помош на Ератостеновото сито.

Задача 6. Напиши програма која претвара број од декаден во бинарен броен систем, и обратно. На пример, $1611_{10} = 11001001011_2$.

Задача 7. За даден почетен природен број a_0 испиши ги броевите добиени според правилото

$$a_{n+1} = \begin{cases} a_n / 2 & , a_n - \text{парен} \\ 3a_n + 1 & , a_n - \text{непарен} \end{cases},\ n = 0,1,2,...$$

сè додека не ја добиеш низата 4, 2, 1 или сè додека не го надминеш дозволениот максимален број членови во низата $n_{\max}$.

5.2. Задачи за самостојна работа

Задача 8. Смести ја во матрица и отпечати ја таблицата за множење од 1 до 10, така да променливата T(i, j) го содржи равенството кое се добива со множење на i и j, на пример T(2, 3) = “2*3=6”.

Задача 9. Одреди го членот од низата $a_1, a_2, ..., a_n$ кој е најмал од сите членови на низата кои се во интервалот (c, d).

Задача 10. За дадена низа нумерички податоци $a_1, a_2, ..., a_n$ одреди ги опсегот, аритметичката средина, модата, медијаната и дисперзијата.

На пример, за низата 5, 2, 6, 2, 1, 4, 5, опсегот е $R = a_{\max} - a_{\min} = 6 - 1 = 5$, аритметичката средина е $av = \frac{1}{7}(5+2+6+2+1+4+5) \approx 3.57$, има две моди, 2 и 5, затоа што тоа се вредности кои се јавуваат најголем број пати, медијана е 4, затоа што тоа е податокот

кој е на средина од подредената низа 1, 2, 2, 4, 5, 5, 6 и дисперзијата е $\text{var} = \frac{1}{7}(5^2 + 2^2 + 6^2 + 2^2 + 1^2 + 4^2 + 5^2) - 3.57^2 \approx 3.11$.

Задача 11. Напиши програма која претвара број од декаден во хексадецимален броен систем, и обратно. На пример, $1611_{10} = 64\text{B}_{16}$.

Задача 12. За даден природен број n пресметај приближно $\sqrt{n}$ со помош на Вавилонскиот метод т.е. почни со произволен природен број x_0 и секој нареден број од низата приближувања пресметувај го според формулата

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{n}{x_k} \right),\ k = 1, 2, 3, \ldots$$

сè додека не се достигне бараната точност т.е. $| x_{k+1} - x_k | < \varepsilon$, за дадено $\varepsilon > 0$ или не се надминат однапред дефинирани максимален број на приближувања $k_{\max}$.

5.3. Задачи со матрици

Задача 13. За матрицата $A = [a_{ij}]_{n \times n}$ најди го збирот на елементи на главната и на споредната дијагонала.

Задача 14. За матрицата $A = [a_{ij}]_{m \times n}$ најди го збирот на елементи на секоја редица и секоја колона.

Задача 15. Дадени се матриците $A = [a_{ij}]_{m \times n}$ и $B = [b_{ij}]_{p \times q}$ и скаларот $\alpha \in \mathbb{R}$. Пресметај $\alpha \cdot A$, $A + B$, $A \cdot B$ и $A^2 - \alpha B$.

Задача 16. Провери дали матрицата $A = [a_{ij}]_{m \times n}$ е ортогонална т.е. дали $A^T \cdot A = E$, каде A^T е транспонираната матрица на матрицата *A*, а E е единичната матрица.

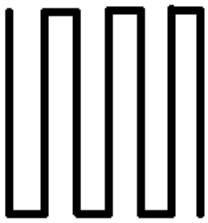
Задача 17. Дадени се матрицата $A = [a_{ij}]_{n \times n}$ и низата $b = (b_1, \ldots, b_n)$.

а) Замени ја k-тата редица на матрицата A со елементите на низата b.
б) Допиши ги елементите на низата b, како последна колона на матрицата A.

Задача 18. Пресметај ја нормата на матрицата $A = [a_{ij}]_{m \times n}$ дефинирана со

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

Задача 19. „Прочитај“ ги елементите на матрицата $A=[a_{ij}]_{m\times n}$ според следната шема (почнувајќи од горниот лев агол):



5.4. Задачи за самостојна работа

Задача 20. Дадени се матриците $A=[a_{ij}]_{n\times n}$ и $B=[b_{ij}]_{n\times n}$. Провери дали матриците комутираат т.е. $A\cdot B=B\cdot A$.

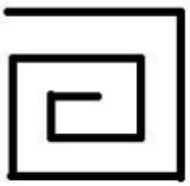
Задача 21. Дадени се матриците $A=[a_{ij}]_{n\times n}$ и $B=[b_{ij}]_{n\times n}$. Провери дали важи равенството $tr(A\cdot B)=tr(B\cdot A)$, каде $tr(M)$ е трага на матрицата *M*, односно збирот на елементите од главната дијагонала.

Задача 22. Провери дали матрицата $A=[a_{ij}]_{m\times n}$ е иденпотентна т.е. дали $A^2=E$.

Задача 23. Пресметај ја нормата на матрицата $A=[a_{ij}]_{m\times n}$ дефинирана со

$$\|A\|=\max_{1\le i\le m,1\le j\le n}\{|a_{ij}|\}.$$

Задача 24. „Прочитај“ ги елементите на матрицата $A=[a_{ij}]_{m\times n}$ според следната шема (почнувајќи од горниот лев агол):



Ирена Стојковска

6. Function и Sub процедури

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
```

Sub процедурата се повикува со реченицата:

```
Call ProcedureName
```

Исто така, Sub процедурата може да зависи и од **параметри** (*par1*, *par2*,…):

```
Private Sub ProcedureName (par1 As Type1, par2 As Type2,…)
        statement(s)
End Sub
```

Во тој случај, Sub процедурата се повикува со задавање на **аргументи**, односно конкретни вредности, изрази или променливи, на местото од параметрите за кои е дефинирана процедурата.

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 стандарден модул како глобални променливи (доколку VB-проектот се состои од повеќе од една форма). Аргументите и параметрите исто така може да се искористат и за пренесување на вредности од Sub процедурата во главната процедура од каде била повикана или во друга Sub процедура.

Пример 1. Најди ги најголемиот и најмалиот елемент на две низи. Дефинирај процедури за најмал, односно најголем елемент од низа.

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 функции.

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 2. Креирај функција за пресметување на 3-ти корен и искористи ја за пресметување на вредноста на изразот

$$A = \sqrt[3]{\frac{2x+y}{x^2-2y^2}}\,.$$

6.1. Задачи

Задача 1. Табелирај ја вредноста на функцијата $f(x)=\frac{\ln(x-1)}{4-x^2}$ на интервалот $[-4,4]$ со чекор $h=0.5$.

Задача 2. Табелирај ја вредноста на функцијата $f(x,y)=\frac{x^2+y}{x-y}+\sqrt{x^2-1}$ на интервалот $[-2,2]\times[-2,2]$ со чекори $h_x=h_y=0.5$.

Задача 3. Со метод на десетично делење, реши ја равенката $x\ln x-1=0$ на интервалот $[1,2]$ и точност $\varepsilon>0$.

Метод на десетично делење. При решавање на равенката $f(x)=0$ со методот на десетично делење, прво се локализира решението, односно се наоѓа интервал $[a,b]$ во кој равенката има единствен корен (решение). Потоа, интервалот $[a,b]$ се дели на 10 подинтервали со еднаква должина $[x_0,x_1],[x_1,x_2],\ldots,[x_9,x_{10}]$, каде $x_i=a+i\cdot\frac{b-a}{10}$, $i=0,1,2,\ldots,10$ и се наоѓа оној подинтервал кој го содржи решението, односно подинтервалот $[x_i,x_{i+1}]$ за кој важи $f(x_i)\cdot f(x_{i+1})<0$. Потоа, тој интервал станува $[a,b]$ и целата постапка на десетично делење на интервалите се повторува сè додека должината на интервалот $b-a\geq\varepsilon$. Кога ќе сопре постапката, за решение на равенката се зема $x^*=\frac{a+b}{2}$.

Задача 4. Приближно пресметај ја вредноста на интегралот $\int_{-1}^{0}\sqrt{4-\sin^2 x}\;dx$, со помош на Риманови суми и точност $\varepsilon>0$.

Риманови суми. Нека $f(x)$ е непрекината функција на интервалот $[a,b]$. Тогаш, приближната вредност на определениот интеграл може да ја пресметаме со Риманови

суми, односно $\int_a^b f(x)\,dx \approx \sum_{k=0}^{n-1} f(u_i)\cdot\Delta x_i$, каде $a = x_0 < x_1 < \ldots < x_n = b$ е разбивање на $[a,b]$, $u_i \in [x_i, x_{i+1}]$ и $\Delta x_i = x_{i+1} - x_i$.

За приближно пресметување со точност $\varepsilon > 0$, се земаат еквидистантни јазли $x_i = a + i\cdot\frac{b-a}{n}$, $i = \overline{0,n}$, при што n се определува така да $\frac{b-a}{n} < \varepsilon$, од каде следи дека $n = [\frac{b-a}{\varepsilon}]+1$, $\Delta x_i = \frac{b-a}{n}$, а за $u_i \in [x_i, x_{i+1}]$ се земаат левите граници на интервалот т.е. $u_i = x_i$. Тогаш, приближната вредност на интегралот е $\int_a^b f(x)\,dx \approx \frac{b-a}{n}\sum_{k=0}^{n-1} f(x_i)$.

6.2. Задачи за самостојна работа

Задача 5. Најди ги двата најголеми елементи на низата a и двата најмали елементи на низата b, со помош на процедура за сортирање на низа.

Задача 6. Креирај функции за пресметување на максумум, односно минимум на два броја и искористи ги за пресметување на вредноста на изразот

$$A = 1 + \min\{x-y, x+y\} + \max^2\left\{\frac{x-y}{x+y}, \frac{x+y}{x-y}\right\}.$$

Задача 7. Со Њутн-Рафсонов метод реши ја равенката $x\ln x - 1 = 0$ на интервалот $[1,2]$ и точност $\varepsilon > 0$.

Њутн-Рафсонов метод. При решавање на равенката $f(x) = 0$ со Њутн-Рафсонов метод, се тргнува од почетно приближување x_0 (кое треба да се внесе) и се креира низа од приближувања според итеративанат формула

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ k = 0,1,2,\ldots$$

сè додека има напредок на алгоритмот, т.е. $|x_{k+1} - x_k| \geq \varepsilon$, каде $\varepsilon > 0$ е дадена точност. Последното приближување се зема за приближно решение на дадената равенка.

Задача 8. Приближно пресметај ја вредноста на интегралот $\int_{-1}^{0} \sqrt{4-\sin^2 x}\,dx$, со помош на Симпсоновата формула и точност $\varepsilon > 0$.

Симсонова формула. Нека $f(x)$ е непрекината функција на интервалот $[a,b]$. Тогаш, приближната вредност на Опеделениот интеграл може да се пресмета според Симпсонова формула, односно

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\cdot\left(f(a) + 4\cdot\sum_{k=1}^{n} f(x_{2k-1}) + 2\cdot\sum_{k=1}^{n-1} f(x_{2k}) + f(b)\right),$$

каде $h = \frac{b-a}{2n}$, $a = x_0 < x_1 < ... < x_{2n-1} < x_{2n} = b$ се еквидистантни јазли на $[a,b]$, т.е. $x_k = a + k \cdot h, \quad k = \overline{0,2n}$.

За приближно пресметување со точност $\varepsilon > 0$, се пресметува сè додека $\frac{|I_1 - I_2|}{2^4 - 1} \geq \varepsilon$ (Рунгеов критериум), каде I_1 и I_2 се две последователни проближни вредности на интегралот добиени за чекори h_1 и $h_2 = \frac{1}{2} h_1$ соодветно, односно при две последователни вредности на n.

Ирена Стојковска

7. Визуелно програмирање во Visual Basic

Визуелното програмирање во Visual Basic подразбира програми кои 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**.

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.

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 PictureBox се: Picture, Font, Visible, …

Почетни имиња кои се доделени на објектите се од обликот Text1, Text2, …, Label1, Label2, …, Command1, Command2, …, Picture1, Picture2,… Но, 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

Преименувањето на објектите не е задолжително, но е препорачливо.

Кога се извршува една Visual Basic програма, на екранот се појавуваат формата и контролите на неа. Ништо не се случува сè додека корисникот не превземе одредено

дејствие, како на пример кликање на некоја конрола. Таквото дејство се нарекува **настан**.

Програмите во Visual Basic се состојат од три дела: **визуелен приказ (interface), почетни вредности на својствата и код**. **Кодот** се состои од реченици (програмски редови) кои извршуваат прецизни задачи за време на извршувањето на програмата. **Вредности на својствата** на обектите може да се доделуваат и преку кодот, а синтаксата за тоа е:

objectName.property = setting

каде *objectName* е името на објектот, *property* е својството, а setting е вредноста која му ја доделуваме. На пример,

```
txtPrezime.Text = ""
cmdPresmetaj.Visible = True
```

Преку кодот може да се повикаат одредени **методи** - дејствиа кои соодветниот објект може да ги извршува

objectName.method [display]

каде *objectName* е името на објектот, *method* е методот, а display е дејствието кое методот треба да го изврши. На пример,

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

Значи, треба да се запазат следните три чекори при креирањето на една Visual Basic порограма:

- 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.

Пример 1. Дизајнирај форма за влез која проверува дали се точно внесени корисничкото име (“admin”) и лозинката (“123456”) и ако е точен влезот, тогаш ја затвара формата за влез и отвара форма за добредојде (“Wellcome!”).

Пример 2. Дизајнирај форма за случаен избор на „петка“ или „глава“. Со кликање на командно копче „ФРЛИ“ случајно се избира една од двете слики и се прикажува во PictureBox.

7.1. Задачи за самостојна работа

Задача 1. Дизајнирај форма за случаен избор на една од шесте страни на една коцка. Со кликање на командно копче „ФРЛИ“ да се прикаже избраната страна во PictureBox.

Задача 2. Дизајнирај тест со 5 случајно избрани задачи со собирање и оземање до 20. По пополнување на тестот, прикажи евалуација на одговорите, кои се точни, а кои не.

Задача 3*. Дизајнирај едноставен калкулатор само за основните операции +, -, *, /. На пример, https://www.online-calculator.com/simple-calculator/.

Ирена Стојковска

8. Графика во Visual Basic

Објекти на кои може да се применат графичките методи се формите (Form) и полињата за слика (PictureBox). Основни чекори при графичките прикази во Visual Basic се: задавање на координатниот систем, користење на графички методи за цртање на линии, правоаголници, точки, кругови (Line, PSet, Circle) и поставување на текст на соодветните места.

8.1. Задавање на координатниот систем

Основното поставување на координатниот систем е со координатниот почеток (0,0) во горниот лев агол на објектот, при што x-оската е насочена од лево на десно, y-оската е насочена од горе на долу. Меѓутоа, со менување на вредностите на својствата ScaleWidth, ScaleHeight, ScaleLeft, ScaleTop на соодветниот објект може да поставиме нов координатен систем. На пример, со програмските редови:

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

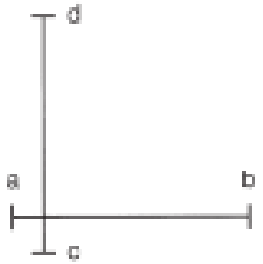
на објектот Picture1 се поставува координатен систем со 30 единици по ширина и 30 единици висина, и координатен почеток на средината на објектот, при што x-оската е насочена од лево на десно, а y-оската е насочена од долу на горе (насоката се менува со знакот -). Истото прилагодување на кординатниот систем се постигнува со Scale методот:

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

или во општ случај со

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

се поставува координатен систем од a до b по x-оската и од c до d по y-оската.



8.2. Графички методи Line, PSet и Circle

Основни графички методи во Visual Basic се Line, PSet и Circle. Отсечка со почеток во точката (x1,y1) и крај во точката (x2,y2) се црта со методот Line со програмскиот ред:

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

Точка со координати (x,y) се нанесува со методот PSet со програмскиот ред:

```
Picture1.PSet (x,y)
```

Кружница со центар во (x,y) и радиус r се црта со методот Circle со програмскиот ред:

```
Picture1.Circle (x,y), r
```

Посложената употреба на графичките методи подразбира користење на боја која се доделува со помош на RGB функцијата, како и дополнителни поставувања за методите Line и Circle да може да се цртаат правоаголници, кружни лаци, кружни исечоци и елипси. Така, правоаголник со дијагонални темиња (x1,y1) и (x2,y2) во избрана боја color се црта со програмскиот ред:

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

На пример, црвен правоаголник со дијагонално спротивни темиња во (2,2) и (6,4) се црта со програмскиот ред:

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

А доколку сакаме да биде целосно обоен тогаш треба да додадеме F, односно:

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

Боја може да се додаде и при цртање на точка со методот PSet, т.е.

```
Picture1.PSet (x,y), color
```

Доколку се работи за точка која е некој од краевите на отсечка или е некое теме од некоја геометриска фигура, тогаш пожелно е со помош на својството DrawWidth да се зададе поголема дебелина на нацртаната точка. Така, зелена точка со координати (-3,2) може да нацртаме со програмските редови:

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

Треба да се внимава, да се врати вредноста на DrawWidth на 1, т.е. нејзината почетна вредност, пред да се продолжи со цртањето.

Методот Circle овозможува цртање на кружни лаци, кружни исечоци и елипси. Со програмскиот ред:

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

се црта дел од елипса со центар (x,y), радиус r (поголемата полуоска т.е. max{a,b}), во боја color, со почеток во start, крај во end и коефициент на сплеснаност aspect (количникот b/a). Доколку не се наведат start, end, aspect се земаат вредностите 0, 2π, 1 соодветно и всушност се добива целосна кружница со центар во (x,y) и радиус r. При користење на некој графички метод, дел од параметрите може да се изостават или прескокнат и во тој случај се земаат нивните почетни поставувања. На пример, сина елипса со центар во (0,0) и полуоски a=2.5 и b=5 се црта со:

```
Picture1.Circle (0,0), 5, RGB(0,0,255), , , 2
```

Или црвен кружен лак т.е. дел од кужница со центар во (0,0) и радиус 5, од аголот Pi/6 до аголот Pi/3 се црта со:

```
Pi = 4*Atn(1)
Picture1.Circle (0,0), 5, RGB(255,0,0), Pi/6 , Pi/3
```

Додека пак црвен кружен исечок т.е. дел од круг со центар во (0,0) и радиус 5, од аголот Pi/6 до аголот Pi/3 се црта со:

```
Pi = 4*Atn(1)
Picture1.Circle (0,0), 5, RGB(255,0,0), - Pi/6 , - Pi/3
```

Знакот – пред големината на аглите означува поврзување на точката од кружницата со нејзиниот центар, а не негативен агол.

Начинот на исполнување на затворените форми нацртани со методот Circle е одреден од вредноста на својството FillStyle, а бојата на исполнување од својството FillColor чии основни поставувања се целосна исполнетост (Solid т.е. 0) и црна боја (т.е. RGB(0,0,0)) соодветно. Така со:

```
Picture1.FillColor = RGB(255,0,0)
Picture1.FillStyle = 3
Pi = 4*Atn(1)
Picture1.Circle (0,0), 5, RGB(255,0,0), - Pi/6 , - Pi/3
```

се црта кружен исечок т.е. дел од круг со центар во (0,0) и радиус 5, од аголот Pi/6 до аголот Pi/3 кој е вертикално шртафиран со црвена боја.

8.3. 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, со програмските редови:

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

ќе се испише текстот "y=0" почнувајќи од точката со координати (13,1). Попрецизно позиционирање на текстот се врши со методите TextHeight и TextWidth. На пример, со програмските редови:

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

текстот “y=0” ќе се позиционира, така да точката (13,1) е на средината на текстот во однос на ширината и висината на текстот.

Пример 1. Нацртај ги на ист координатен систем и означи ги:

а) црвена отсечка $AB / A(4,5), B(10,13) /$,

б) син правоаголник $CDEF / C(-10,-10), D(-5,-10), E(-5,-2), F(-10,-2) /$,

в) обоен зелен квадрат со дијагонално поставени темиња во точките $(5,-5)$ и $(8,-2)$,

г) виолетова кружница со центар во $(-10,10)$ и радиус 3,

д) црна елипса обоена со жолта боја со центар во $(-2,6)$ и полуоски $a=4, b=2$.

8.4. Задачи

Задача 1. Нацртај ги графиците на следните функции:

1) кружница $x^2+y^2+2x-6y-6=0$,

2) елипса $9x^2+y^2-36x+2y+28=0$,

3) $y=\dfrac{x^3}{4(2-x)^2}$,

4) $y=\sqrt{\dfrac{x^3-2x^2}{x-3}}$,

5) $x=\dfrac{t^2}{4(1-t)}$, $y=\dfrac{t^3}{8(t-1)}$,

6) $\rho=\cos(3\varphi)$,

7) кардиоида $\rho=1+a\cos(\varphi),\ a>0$,

8) Декартов лист $x^3+y^3=3xy$.

Задача 2. Најди го геометриското место на точки на центрите на кружниците кои минуваат низ точката $A(a_1,a_2)$ и на кои x-оската им е тангента.

Задача 3. Најди го геометриското место на точки на средините на отсечките со една крајна точка во точката $A(a_1,a_2)$ и другата се движи по параболата $y=a(x-x_0)^2+c$.

Задача 4*. Прикажи ги графички резултатите од едно гласање (ЗА, ПРОТИВ, ВОЗДРЖАНИ) на столбест дијаграм (bar chart) и секторски дијаграм (pie chart), при влезни податоци: вкупен број на гласачи, број на гласачи кои гласале ЗА и број на гласачи кои гласале ПРОТИВ.

8.5. Задачи за самостојна работа

Задача 5. Нацртај ги графиците на следните функции:

1) $(x-5)^2+(y-3)^2=36$

2) $\dfrac{(x+3)^2}{4}+\dfrac{(y-5)^2}{16}=1$

3) $y=\dfrac{x^2+3x}{x^2-4}$

4) $x=t^2,\ y=\dfrac{1}{2}t^3$

5) $\rho=\sin(2\varphi)$

6) $x^2y^2=x^3-y^3$.

Задача 6. Најди го геометриското место на точки на тежиштата на триаголниците со две темиња во точките $A(a_1,a_2)$ и $B(b_1,b_2)$ и третото се движи по x-оската.

Задача 7. Даден е триаголникот $ABC / A(a_1,a_2), B(b_1,b_2), C(c_1,c_2) /$. Нацртај ги опишаната и впишаната кружница на триаголникот ABC .

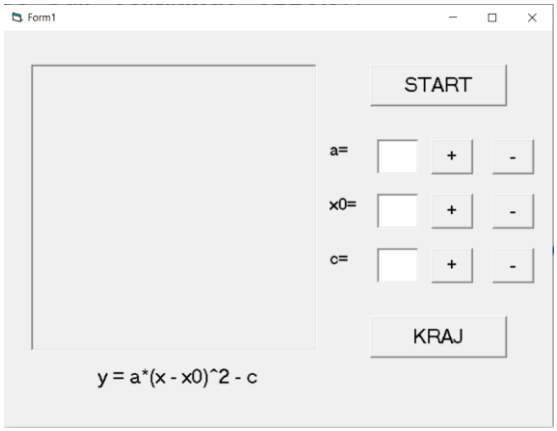
Задача 8. Прикажи ги на столбест дијаграм податоците за избор на омилено хоби меѓу одредена група испитаници дадени во следната табела:

	Готвење	**Трчање**	**Возење велосипед**	**Читање книги**
Мажи	10	25	10	20
Жени	20	30	5	40

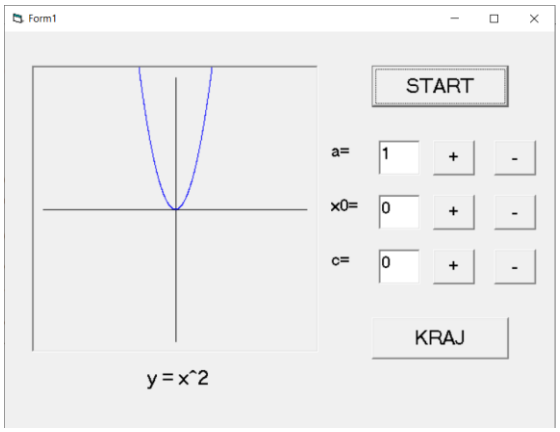
Податоците прикажи ги со два типа споредбени столбести дијаграми – групиран столбест дијаграм (grouped bar chart) и нареден столбест дијаграм (stacked bar chart).
https://www.statisticshowto.com/probability-and-statistics/descriptive-statistics/bar-chart-bar-graph-examples/

Задача 9*. Прикажи ги графички со столбест дијаграм (bar chart) исходите од n фрлања на фер коцка за играње.

Задача 10*. Прикажи го нагледно менувањето на графикот на параболата $y = a(x - x_0)^2 + c$, во зависност од промената на вредноста на параметрите a, x_0 и c. Кога ќе се стартува програмата (со кликање на копчето START), на цртежот да се нацрта параболата за $a = 1, x_0 = 0, c = 0$, а за секој параметар да има копчиња за зголемување и намалување на неговата вредност, така да при секоја промена на вредноста на параметарот на цртежот прво да се избрише претходно нацртаната парабола, а потоа да се исцрта соодветната парабола во зависност од вредноста на параметрите. Предлог интерфејс:



Изглед по стартување на програмата



Изглед по кликање на копчето START

Дополнително: Означи го координатниот систем (координатните оски и големината на единичната отсечка). Расмисли дали ќе дозволиш неограничено менување на вредностите на параметрите и како во тој случај ќе биде прикажан делот од графикот каде се наоѓа параболата или ќе се одлучиш за ограничено менување на вредностите на параметрите.

Ирена Стојковска

9. Работа со датотеки во Visual Basic

Датотеките се состојат од редови, односно записи или рекорди 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, поени од испит).

9.1. Кориснички дефиниран тип на податоци

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

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, за информации за еден студент (име, презиме, број на индекс, насока) дефинираме нов тип податоци Student:

Type *Student*
    Ime **As** *String*
    Prezime **As** *String*
    BrNaIndeks **As** *Long*
    Nasoka **As** *String*
End Type

Дефинирањето на корисничкиот тип на податоци се извршува или во General Declarations, ако VB проектот се состои од една форма или во модул кој се додава со Project → Add Module → Module, ако VB проектот се состои од две или повеќе форми.

9.2. Секвенционални датотеки (Sequential Files)

За да се пристапи кон една секвенционална датотека таа треба да се отвори со:

Open "filespec" **For** mode **As** #n

каде “filespec” е патеката по која се пристапува до датотеката, ако таа е во фолдерот со останатите VB фајлови од проектот тогаш се користи App.Path & “/filename” каде “/filename” е името на датотеката заедно со нејзината екстензија. Потоа, mode е начинот на пристап кон датотеката. Така, начинот Output се користи за креирање на датотеката и за внесување податоци во неа од нејзиниот почеток. Начинот Append се користи за отварање на постоечка датотека за допишување на податоци во датотеката на крајот од датотеката, а ако датотеката не постои со Append се креира. Начинот Input се користи за отварање на датотеката за читање на податоци од неа. Датотеката отворена за

запишување (Output или Append) треба прво да се затвори за да може потоа да се отвори за читање (Input). Бројот n е број од 1 до 511 и се користи како референтен број за време на работа со датотеката отворена со тој број. На пример со:

Open App.Path & "\spisok.dat" **For** Output **As** #1

Во папката на VB проектот се креира датотека “spisok.dat” и се отвара за запишување на податоци под референтен број 1.

При отварање на датотеката за читање (Input), доколку таа не постои, ќе резултира со стопирање на програмата со порака за грешка “File not found”. Со функцијата

Dir (“filespec”)

се проверува дали датотеката е веќе креирана, ако вредноста на оваа функција е празен стринг “”, тогаш конкретната датотека не постои, а ако постои, тогаш вредноста на функцијата ќе биде името на датотеката.

Запишувањето и читањето на податоци во, односно од, датотеката се изведува ред по ред т.е. одеднаш се запишува, односно чита, целиот запис (рекорд). Синтаксата за запишување на податоците во датотеката отворена под референтниот број n е:

Write #n, *varlist*

каде *varlist* е листа од податоци или променливи (а може да биде и само еден податок, односно само една променлива). Синтаксата за читање на податоците од датотеката отворена под референтен број n е:

Input #n, *varlist*

каде *varlist* е листа од податоци или променливи (а може да биде и само еден податок, односно само една променлива). Важно е да се запомни дека секој ред од една датотека се состои од податоци, или листи од податоци, од ист тип. На пример, доколку го дефинираме користичкиот дефиниран тип на податоци Student како во претходниот дел, тогаш запишувањето на еден запис во датотеката “spisok.dat” отворена под референтен број 1 може да се изврши со:

Write #1, “Ana”, “Mihajlovska”, 15432, “Nastavna matematika”

или ако податоците се превземаат од текстуални полиња, запишувањето може да се изврши на следниот начин:

```
Dim record As Student
record.Ime = Text1.Text
record.Prezime = Text2.Text
record.BrNaIndeks = Val(Text3.Text)
record.Nasoka = Text4.Text
Write #1, record.Ime, record.Prezime, record.BrNaIndeks, record.Nasoka
```

Бидејќи читањето на податоците се изведува ред по ред, се користи булеан функцијата **EOF**(n), кратенка од Enf Of File, со која се проверува дали е дојден крајот на датотеката отворена под референтен број n.

По завршување со работата со датотеката, таа треба да се затвори, пред да се отвори за друга намена. Датотеката отворена под референтен број n се затвора со:

Close #n

Со оваа наредба се прекинува комуникацискатa врска меѓу датотеката и референтниот број n, кој потоа може да се додели на друга датотека. Една датотека не може истовремено да биде отворена повеќе од еднаш, но повеќе датотеки може да бидат отворени истовремено, на пример кога од една датотека се читаат податоци и тие се запишуваат во друга датотека. Со наредбата **Close** (без наведување на референтен број) се затвараат сите моментално отворени датотеки.

Кај секвенционалните датотеки не може директно да се измени или избрише еден конкретен запис, мора да се креира нова датотека со читање и запишување запис по запис од старата во новата датотека. Потоа, откако ќе се затворат датотеките, старата датотека се брише со:

Kill *“oldfilespec”*

А новата датотека се преименува со името на старата датотека со:

Name *“newfilespec”* **As** *“oldfilespec”*

Секвенционалните датотеки ефикасно го користат меморискиот простор на дискот и се лесни за креирање и работа со нив, имено нивната содржина може да се менува и во некој текст едитор, на пример Notepad. Нивен главен недостаток е читањето на голем дел од содржината за да се пристапи кон некој одреден запис, односно податок, а и неможноста за директно менување или отстранување на конкретен запис.

Пример 1. Креирај секвенционална датотека со податоци за студенти (име, презиме, број на индекс, поени од испит).
а) Внеси податоци за неколку студенти и излистај ги.
б) Излистај ги студентите со број на индекс поголем од …
в) Излистај ги положените студенти, оние со 50 и повеќе поени (креирај нова датотека) и пресметај просек на положените.
г) Сортирај ги студентите според поените, а второстепено според презимето.
д) Отстрани запис.
ѓ) Измени запис.
е*) Прикажи ги резултатите од испитот графички со столбест дијаграм.

Забелешка: За листање на податоците од датотеката користи ја контролата ListBox. Додавање на записи во List1 (објект на формата нанесен со контролата ListBox) се извршува со методот AddItem. Бришење на содржината на List1 се извршува со методот Clear.

9.3. Датотеки со случаен пристап (Random Access Files)

Датотеките со случаен пристап се како низи од записи нумерирани со 1, 2, 3, и.т.н., и кон нив може да се пристапи преку нивниот реден број. Отварањето на датотека со случаен пристап е со наредбата:

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

каде "filespec" е патеката по која се пристапува до датотеката (ако датотеката е снимена во папката на VB проектот може да се кортисти App.Path & "/filename"), Random е начинот на отварање на датотеката со случаен пристап (во отворените датотеки со случаен пристап, може истовремено и да се запишуваат и да се читаат податоци), ако датотеката не постои, првиот пат кога ќе се отвори, таа и се креира, n е референтниот број (од 1 до 511) под кој се отвара датотеката, а Len е должината на еден запис (*recvar* е променлива од типот податоци од кој се записите во датотеката). За датотеките со случаен пристап задолжително (!) треба текстуалните променливи да бидат декларирани како стрингови со фиксна должина т.е.

Dim *var* **As** String *m

каде m е должината на текстуалниот податок складиран во променливата *var*.

Запишување на податоците од променливата recvar (која треба да биде променлива од корисничкиот дефиниран тип на податоци за еден запис во датотеката), како запис со реден број br, во датотеката отворена под референтен број n, се извршува со:

Put #n, br, recvar

Читање на податоците од записот со реден број br, од датотеката отворена под референтен број n и доделување на прочитаната вредност на променливата recvar се извршува со:

Get #n, br, recvar

Вкупниот број на карактери во датотеката отворена под референтен број n се пресметува со функцијата **LOF**(n), кратенка од Length Of File, па така, знаејќи дека бројот на карактери на еден запис е Len(recvar), за вкупниот број на записи имаме дека е:

NumberOfRecords = **LOF**(n) / Len(recvar)

Затворањето на датотеката отворена под референтен број n се извршува со:

Close #n

Со наредбата **Close** (без наведување на референтен број) се затвараат сите моментално отворени датотеки. Исто како и кај секвенционалните датотеки, бришењето на датотека, односно преименување се извршува со **Kill**, односно **Name As**.

Датотеките со случаен пристап се познати и како датотеки со директен или релативен пристап. Бидејќи секој запис има еднаков број на карактери, комјутерот може да пресмета каде може да го најде записот, па затоа пребарувањето не е секвенционално.

За разлика од секвенционалните датотеки, датотеките со случаен пристап не мора да се затворат меѓу дејствијата запишување и читање на податоци. Записите во датотеките со случаен пристап не мора да се пополнуваат последователно, кон секој запис може да се пристапи со помош на неговиот реден број.

Одлуката за избор меѓу секвенционална и датотека со случаен пристап зависи од начинот на обработка на податоците за решавање на конкретниот проблем. Ако обработката подразбира поминување низ сите податоци, тогаш може да се работи со секвенционални датотеки. Ако обработката поразбира барање на конкретен податок, тогаш датотеките со случаен пристап се препорачливи.

Една датотека креирана како секвенционална датотека, односно датотека со случаен пристап, не може да премине во датотека со случаен пристап, односно секвенционална датотека.

Пример 2. Креирај датотека со случаен пристап со податоци за држави (име на државата, главен град, површина (во km^2), број на жители (во 1000)).
а) Внеси податоци за неколку држави и излистај ги.
б) Излистај ги државите со главен град на буквата ...
в) Излистај ги државите со површина од ... до ...
г) Излистај ги државите според даден клучен збор ...
д) Најди ја нагусто населената држава.
ѓ) Сортирај ги државите според името, површината, бројот на жители.
е) Отстрани запис.
ж) Измени запис.
з*) Прикажи ја графички врската меѓу површината и бројот на жители (scatter plot т.е. точкест дијаграм).

Забелешка: За листање на податоците од датотеката користи ја контролата ListBox.

9.4. Задачи за самостојна работа

Задача 1. Креирај датотека со податоци за книги во една библиотека (автор на книгата, наслов, издавач, година на издавање, број на примероци).
а) Внеси податоци за неколку книги и излистај ги.
б) Излистај ги книгите од авторот ...
в) Излистај ги книгите со година на издавање од ... до ...
г) Излистај ги книгите според даден клучен збор ...
д) Најди ја најстарата книга (или најстарите книги ако се повеќе од една).
ѓ) Сортирај ги книгите според авторот, а второстепено според годината на издавање.
е) Сортирај ги книгите според издавачот, второстепено според авторот, а третостепено според годината на издавање.
ж) Отстрани запис.
з) Измени запис.
ѕ*) Реализирај земање/враќање на книга/книги од библиотеката.

Задача 2. Креирај датотека со податоци за студенти на online курс (име, презиме, пол, возраст, електронска адреса, дали претходно посетувал online настава организирана од истиот едукативен центар).

а) Внеси податоци за неколку студенти и излистај ги.

б) Излистај ги студентите кои не посетувале претходно online настава организирана од истиот едукативен центар.

в) Излистај ги студентите на возраст од ... до ...

г) Излистај ги студентите според даден клучен збор ...

д) Најди го највозрасниот студент (или студенти ако се повеќе од еден).

ѓ) Сортирај ги студентите според презимето, полот, возраста.

е) Отстрани запис.

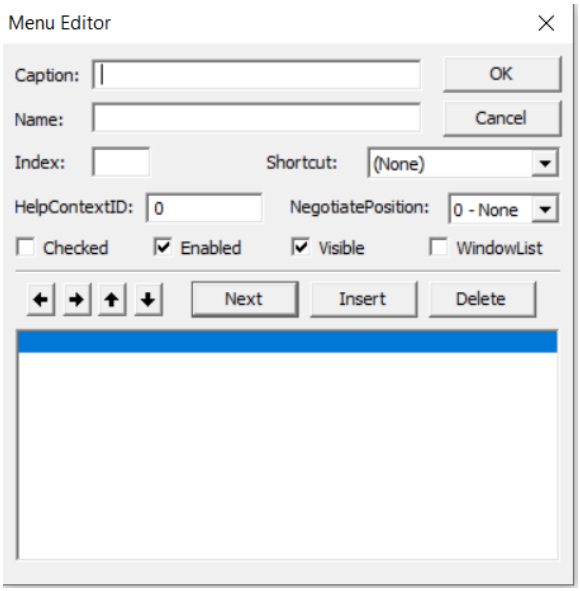
ж) Измени запис.

з*) Прикажи ги графички половата и возрасната распределба на студентите на online курсот.

9.5. Додавање на мени и други контоли

За да се добие покомпактен изглед на визуелниот приказ (интерфејсот) на VB проектот може да се додаде мени (Menu). Менито може да се додаде на секоја форма. Единствен настан на мени контролите е настанот Click, па затоа се смета дека една мени контрола е замена за командно копче. Но, не секогаш е пожелно додавањето на мени, на пример нема потреба додавење на мени, ако на формата има само 2-3 командни копчиња.

Менито се вметнува со Tools → Menu Editor… при што се отвара мени едиторот со кој се уредува содржината на менито. Менито може да содржи повеќе ставки, а секоја ставка може да содржи подставки, односно да биде подмени на главното мени.



Со контролата Option Button се нанесува кружно копче за избор на опција. Текстот кој се запишува на него е со помош на својството Caption, а дали е селектирано или не покажува вредноста на својството Value, кое може да прима две вредности True или False. Кога на формата има повеќе од едно копче за избор, само едно може да биде селектирано.

Со конторолата Check Box се нанесува квадратно копче за потврдување. На него текст се запишува со помош на својството Caption, а неговото својство Value покажува дали не е селектирано (0 – Unckecked), селектирано (1 – Checked) или посивено (2 – Grayed). Ако на фомрата има повеќе копчиња за потврдување, може да бидат селектирани и повеќе од едно.

До сега видовме дека со контолата List Box се нанесува поле за листа во кое со методот AddItem се додаваат записи, притоа еден запис т.е. еден ред во листата е еден текстуален податок. Корисно е да се знае дека, кон податоците запишани во една листа може и директно да се пристапи. Имено, редовите во листата се нумерирани почнувајќи од 0. Така, кон вредноста на i-тиот ред на List1 се пристапува со List1.List(i-1). Проверката дали i-тиот ред на List1 е селектиран се извршува со List1.Selected(i-1) = True. Па, со програмските редови

```
If List1.Selected(i - 1) = True Then
        zapis = List1.List(i - 1)
End If
```

на променливата zapis ѝ се доделува вредноста на i-тиот ред од List1, ако тој е селектиран. Вкупниот број на редови во листата List1 е List1.ListCount. Содржината на листата се брише со методот Clear.

Со контролата Combo Box се нанесува комбо листа. Текстот на него се запишува со својството Text, а записи во неговата листа, која се отвара со кликање на стрелката, се додаваат со AddItem. Настанот Click се реализира кога за време на извршување на прогамата се одбере некој запис од комбо листата.

Постојат и други контроли со кои може да се нанесат друг вид на објекти на формите. Изборот на конторолата зависи од планираниот дизајн на проектот, кој пак зависи од намената на проектот. Еден проблем секогаш може да се реши на различни начини, па затоа може да се дизајнираат различни проекти за иста намена. Но, најважно е во некој момент да почнете да правите сопствени програми за сопствени проекти, бидејќи така најмногу се учи и на тој начин ќе почувствувате најголемо задоволство.

„Секој треба да научи да програмира на компјутер, бидејќи така се учи како се размислува.“ – Стив Џобс

Пример 3. Креирај VB проект со податоци за растенија во еден градинарски магацин (име на растение, категорија (зеленчук/овошје/друго), површина одредена за чување на тој вид растение (во m^2), присутен број на корени, дозволен максимален број на корени). Проектот треба да обезбеди внесување на нови записи, листање на сите записи, листање според одреден критериум (зеленчук, овошје, друго, клучен збор), сортирање според одреден критериум (име, површина на магацинскиот дел, број на присутни корени), наоѓање на растение кое задоволува одреден услов (најзастапено, најгусто сместено), бришење или менување на постоечки записи (пристапот кон саканиот запис да е со селектирање на запис излистан во листа). Исто така треба да може да се реализира одлив / дотур на садници во градинарскиот магацин (без пристап кон менување на сите податоци за еден запис, туку само бројот на корени).

Забелешка: Користи мени и други контроли за извршување на барањата.

9.6. Задачи за самостојна работа (Идеи за проекти)

Задача 3. Креирај VB проект со податоци за **кино проекции** (кино сала, филм, жанр, почеток на филмот, времетраење на филмот, број на слободни седишта). Програмата покрај обичен пристап, да има и кориснички и администраторски пристап:

Обичниот пристап е само за пребарување на филмските проекции.

Корисникот покрај пребарување на филмовите, треба да може и да резервира одреден број на седишта за конкретна проекција на филм.

Администраторот треба да има пристап кон истите опции како корисникот и дополнително да може да внесува нови записи, да ги брише или да внесува измени во постоечките записи.

И корисникот и администраторот имаат свои кориснички имиња и лозинки. Администраторското корисничко име и лозинка се внесуваат во кодот на програмата, а корисничките имиња и лозинки се внесуваат при регистрација на новиот корисник и се чуваат во посебна датотека, за да му се овозможи пристап на веќе регистриран корисник.

Задача 4. Креирај VB проект со податоци за **online продавница** за облека (категорија (машка/женска/детска), вид (блуза/панталони/обувки/друго), единствен код, количина од секоја големина, цена на едно парче). Програмата покрај обичен пристап, да има и кориснички и администраторски пристап:

Обичниот пристап е само за пребарување на артиклите.

Корисникот покрај пребарување на артиклите, треба да може и да ја наполни кошничката и да ги купи избраните артикли.

Администраторот треба да има пристап кон истите опции како корисникот и дополнително да може да внесува нови записи, да ги брише или да внесува измени во постоечките записи.

И корисникот и администраторот имаат свои кориснички имиња и лозинки. Администраторското корисничко име и лозинка се внесуваат во кодот на програмата, а корисничките имиња и лозинки се внесуваат при регистрација на новиот корисник и се чуваат во посебна датотека, за да му се овозможи пристап на веќе регистриран корисник.

Задача 5. Креирај VB проект за **online курс** (теми, лекции, решени примери, задачи за решавање, тестови). Програмата покрај обичен пристап, да има и кориснички и администраторски пристап:

Обичниот пристап е само за читање на лекциите со примерите и задачите, без пристап до тестовите.

Корисникот покрај читањето на лекциите со примерите и задачите ила и пристап до тестовите. Тестот се состои од одреден број случајно одбрани 10 различни прашања/задачи од база на прашања/задачи. По решавањето на тестот корисникот добива повратна информација дали точно одговорил на прашањата/задачите.

Администраторот треба да има пристап кон истите опции како корисникот и дополнително да може да внесува нови теми, лекции, решени примери, задачи за решавање, прашања и задачи за тестовите, да ги брише или да внесува измени во постоечките записи.

И корисникот и администраторот имаат свои кориснички имиња и лозинки. Администраторското корисничко име и лозинка се внесуваат во кодот на програмата, а корисничките имиња и лозинки се внесуваат при регистрација на новиот корисник и се чуваат во посебна датотека, за да му се овозможи пристап на веќе регистриран корисник.