
3 VIZUĀLIE UN NEVIZUĀLIE KOMPONENTI, LIETOJUMPROGRAMMAS UZSTĀDĪJUMI

Šī laboratorijas darba mērķis ir apskatīties papildus vizuālās komponentes (kontroles) un nostiprināt to pielietošanas scenārijus.

Padoms: *Laboratorijas darba izpildi var uzsākt uzreiz no sadaļas 3.3, atgriežoties pie apraksta sadaļām tikai tad, ja ir neskaidrības uzdevumu izpildīšanā.*

3.1 PAPILDUS VIZUĀLO KOMPONENTU APSKATS

Kā jau tika minēts iepriekšējā laboratorijas darbā – komponenti iedalās vizuālajos (kontrolēs) un nevizuālajos komponentos. Visi vizuālie komponenti ir klases **Control** tiešas vai netiešas apakšklases, tāpēc tās manto visus klases **Control** īpašības, metodes un notikumus.

Apskatīsim vēl vairākas visbiežāk izmantojamās kontroles, kaut arī daļa no tām jau tika praktiski izmantota iepriekšējos laboratorijas darbos.

Label – iezīme.

Mantošanas hierarhija: *Label > Control*

Izmanto, lai uz formas noteiktā vietā un fontā izvadītu tekstuālu informāciju, kuru lietotājs var tikai vizuāli nolasīt, bet ne izvēlēties vai modificēt.

Kontrole **Label** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
AutoEllipsis	Atgriež/uzstāda daudzpunktu rādīšanu iezīmes teksta beigās, ja viss iezīmes teksts nevar tikt izvadīts kontroles izmēru ierobežojumu dēļ.
AutoSize	Atgriež/uzstāda pazīmi, vai kontrole automātiski maina tās vizuālo platumu, lai pilnībā varētu izvadīt tās tekstu.
BorderStyle	Atgriež/uzstāda iezīmes malu stilu.
Image	Atgriež/uzstāda attēlu, kuru rādīt iezīmes fonā.
Text	Atgriež/uzstāda uz iezīmes attēlojamo tekstu.
TextAlign	Atgriež/uzstāda teksta izlīdzināšanas vietu iezīmes robežās. Šai īpašībai ir nozīme tikai tad, ja īpašības AutoSize vērtība ir false .
UseMnemonic	Atgriež/uzstāda mnemoniskas atbalstu. Ja lietotājs nospiež klaviatūras taustiņu kombināciju ALT+<mnemoniskais simbols>, fokuss automātiski pāriet uz nākošo komponentu tabulāciju secībā (t.i. uz kontroli, kuras īpašības TabIndex vērtība ir par vienu lielāka nekā pašas iezīmes TabIndex vērtība). Piemēram, ja tā ir teksta elementa TextBox kontrole, tad kursors tiks aktivizēts tajā.

Kaut arī šī kontrole spēj tikai attēlot informāciju, tomēr tā var reaģēt arī uz vairākiem notikumiem, starp kuriem ir no klases **Control** mantotais **Click**.

LinkLabel – iezīme ar saiti.

Mantošanas hierarhija: *LinkLabel > Label > Control*

Līdzīgi kā iezīmes kontrole – iezīme ar saiti atspoguļo tekstuālu informāciju, bet ar papildus iespēju vizuāli izcelt teksta daļu (vai daļas), uz kuras uzklikšķinot tiek aktivizēts notikums **LinkClicked**, kuru apstrādājot var realizēt reakciju uz saites izvēli.

Kontrole **LinkLabel** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
ActiveLinkColor, LinkColor, DisabledLinkColor, VisitedLinkColor	Atgriež/uzstāda saites teksta krāsas dažādos tā režīmos.
LinkArea	Atgriež/uzstāda to iezīmes teksta daļu, kas tiks attēlota kā saite iezīmes tekstā. Pēc noklusēšanas par saiti tiks izmantots viss iezīmes teksts.
LinkBehavior	Atgriež/uzstāda saites vizuālā izcēluma (t.i. pasvītrojuma) attēlošanas veidu.
Links	Atgriež iezīmē definēto saišu kolekciju.
LinkVisited	Atgriež/uzstāda pazīmi, ka saite ir bijusi noklikšķināta.

Iezīmes ar saiti notikuma **LinkClicked** apstrādātājs kā parametru saņem tipa **LinkLabelLinkClickedEventArgs** instanci. Šī instance satur atsauci uz klases **Link** instanci, kuru var izmantot, lai noskaidrotu, kura tieši saite ir izvēlēta. Bet, uzstādot **Link** instances īpašību **Visited** uz **true**, tiek vizuāli norādīts, ka lietotājs saiti ir apmeklējis.

Kaut arī vizuāli saite izskatās līdzīgi kā tīmekļa pārlūkprogrammās, tomēr reakciju uz saites izvēli nosaka tikai un vienīgi notikuma apstrādātājs – tā nav piesaistīta vienīgi URL saitēm. Piemēram, lai saites izvēlei piesaistītu URL saites atvēršanu, notikuma **LinkClicked** apstrādātājā var realizēt šādu pirmkodu, kas atvērs lietotāja noklusēto pārlūkprogrammu un parādīs norādīto saiti:




```
System.Diagnostics.Process.Start("http://www.rtu.lv");
```

CheckBox – izvēles rūtiņa.

Mantošanas hierarhija: *CheckBox* > *ButtonBase* > *Control*

No mantošanas hierarhijas redzams, ka izvēles rūtiņa ir pogas paveids, un parasti tiek izmantota, lai ļautu uzstādīt vai noņemt kādas bināras pazīmes vērtību.

Kontrole **CheckBox** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
Appearance	Normal – ja kontroli jāatspoguļo ar izvēles rūtiņu un tekstu tai blakām. Button – ja kontroli jāatspoguļo kā pogu ar tekstu, kura var atrasties nospiebtā stāvoklī.
AutoCheck	Atgriež/uzstāda vai izmainīt izvēles rūtiņas stāvokli arī pie peles noklikšķināšanas. Ja false , tad jārealizē notikuma Click apstrādātājs, kurā jārealizē stāvokļu maiņas algoritms.
Checked	Atgriež/uzstāda vai binārās pazīmes vērtība ir uzstādīta (true), vai noņemta (false).
CheckState	Izvēles rūtiņa var atrasties arī trīs stāvokļos: <ul style="list-style-type: none"> Checked (vizuāli kā ) – ja īpašība Checked ir true; Unchecked (vizuāli kā ) – ja īpašība Checked ir false; vai Indeterminate (vizuāli kā ) – nenoteikts, kas atspoguļo stāvokli, kad izvēles rūtiņa ir izvēlēta un ir iekrāsota. Šis stāvoklis ir pieejams tikai tad, ja īpašība ThreeState ir uzstādīta uz true.
ThreeState	Ja uzstādīts uz false , tad izvēles rūtiņa nevar atrasties CheckState stāvoklī

	Indeterminate , ja true – tad var.
CheckAlign	Nosaka izvēles rūtiņas atspoguļošanas vietu attiecībā pret tās tekstu.

Notikumi, kuriem visbiežāk šai kontrolei realizē apstrādātājus:

- **CheckedChanged** – iestājas, kad izmainās īpašības **Checked** vērtība;
- **CheckStateChanged** – iestājas, kad izmainās īpašības **CheckState** vērtība;
- **Click** – iestājas, kad uz kontroles veic peles klikšķi.

ComboBox – kombinētais saraksts.

Mantošanas hierarhija: *ComboBox* > *ListControl* > *Control*


Kombinētā saraksta kontroli lieto, kad nepieciešams no saraksta izvēlēties vienu elementu, ar iespēju norādīt arī sarakstā neesošu elementu. Kombinētais saraksts ir klases **ListControl** pēctecis.

Klase **ListControl** realizē vispārēju saraksta tipa kontroļu funkcionalitāti, definējot šādus būtiskākās īpašības un metodes:

Īpašība/Metode()	Apraksts
FormatString	Atgriež/uzstāda saraksta elementu attēlošanas formātu.
SelectedIndex	Atgriež/uzstāda sarakstā izvēlēta elementa kārtas numurs. Numerācija sākas no 0, bet ar -1 apzīmē to, ka nekas nav izvēlēts.
SelectedValue	Atgriež/uzstāda izvēlēto elementu pēc tā vērtības. Vērtība null norāda uz to, ka neviens elements nav izvēlēts.
GetItemText()	Atgriež parametrā norādītā elementa atspoguļojumu teksta veidā.

Savukārt kontrole **ComboBox** definē šādus papildus īpašības un metodes:

Īpašība/Metode()	Apraksts
AutoCompleteCustomSource	Atgriež/uzstāda automātiskās teksta pabeigšanas teksta virkņu objektu.
AutoCompleteMode	Atgriež/uzstāda automātiskās teksta pabeigšanas režīmu: <ul style="list-style-type: none"> • None – automātiskā teksta pabeigšana ir atslēgta; • Suggest – ieteikumu saraksta parādīšana; • Append – pievienot jau ievadītā teksta beigās ticamāko atlikušā teksta daļu; • SuggestAppend – iepriekšējo divu kombinācija.
AutoCompleteSource	Automātiskās pabeigšanas tekstu avots. Starp iespējamiem avotiem None atslēdz šo iespēju, bet CustomSource gadījumā par avotu tiek izmantota īpašības AutoCompleteCustomSource vērtība, bet ListItems – pašā saraksta elementi.
DrawMode	Saraksta elementu atspoguļošanas režīms: <ul style="list-style-type: none"> • Normal – saraksta elementus atspoguļošanu veiks sistēma un visi saraksta elementi būs ar vienādu augstumu; • OwnerDrawFixed – saraksta elementus būs jāatspoguļo pašai

	<p>lietojumprogrammai un visi saraksta elementi būs ar vienādu augstumu;</p> <ul style="list-style-type: none"> • OwnerDrawVariable – saraksta elementus būs jāatspoguļo pašai lietojumprogrammai, bet saraksta elementi varēs būt ar dažādiem augstumiem. <p>Pēdējos divos režīmos ir jārealizē notikums DrawItem, kas tiek izsaukts katra saraksta elementa atspoguļošanai.</p>
DropDownHeight, DropDownWidth	Atgriež/uzstāda kombinētā saraksta augstumu un platumu pikseļos.
DropDownStyle	<p>Atgriež/uzstāda kombinētā saraksta kontroles stilu:</p> <ul style="list-style-type: none"> • Simple – kombinētā saraksta teksta lauka daļa ir rediģējama, bet saraksta daļa ir redzama visu laiku; • DropDown – teksta lauka daļa ir redzama visu laiku, bet saraksta daļas parādīšanai jānospiež teksta lauka beigās attēlotā poga ; • DropDownList – teksta lauka daļu rediģēt nav iespējams, lai norādītu vērtību, tā jāizvēlas no saraksta daļas, kuru izritina nospiežot teksta lauka beigās attēloto pogu.
DroppedDown	Atgriež/uzstāda pazīmi, vai rādīt kontroles saraksta daļu.
Items	Atgriež/uzstāda saraksta elementus.
MaxDropDownItems	Atgriež/uzstāda maksimālo saraksta elementu skaitu, kas nosaka, cik daudz saraksta elementus vienlaicīgi atspoguļot bez ritināšanas nepieciešamības.
MaxLength	Atgriež/uzstāda maksimālo rakstzīmju skaitu, ko var ievadīt teksta lauka daļā.
SelectedIndex	Atgriež/uzstāda izvēlēta saraksta elementa indeksu.
SelectedItem	Atgriež/uzstāda izvēlēto saraksta elementu, vai null , ja neviens elements nav izvēlēts.
SelectedText, SelectionLength, SelectionStart, Select(), SelectAll()	Atgriež/uzstāda teksta laukā izvēlēta teksta fragmenta, garuma un sākuma pozīciju.
Sorted	Atgriež/uzstāda pazīmi, ka kontroles saraksta daļas elementus automātiski sakārtot vai nē.
BeginUpdate(), EndUpdate()	Atslēdz un atkārtoti ieslēdz saraksta elementu pārzīmēšanu – izmanto, ja saraksta elementi tiek pievienoti pa vienam – pirms pievienošanas atslēdz pārzīmēšanu izsaucot BeginUpdate() , bet visu nepieciešamo elementu pievienošanas beigās, ieslēdz pārzīmēšanu izsaucot EndUpdate() .
FindString(), FindStringExact()	Teksta meklēšana saraksta elementos un pirmā atrastā elementa indeksa atgriešana. Atgriež -1, ja meklētais teksts nav atrasts nevienā saraksta elementā.

Būtiskākie notikumi, kuriem nākas realizēt notikumu apstrādātājus:

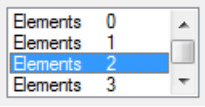
- **SelectedIndexChanged** – ja nomainījās indekss izvēlētajam saraksta elementam;
- **SelectionChangeCommitted** – ja izvēlētais saraksta elements ir atspoguļots teksta lauka daļā un saraksta daļa ir aizvērta.

ListBox – sarakstlodziņš.

Mantošanas hierarhija: *ListBox* > *ListControl* > *Control*

Kontrole **ListBox** tāpat kā **ComboBox** ir klases **ListControl** pēctecis, tāpēc definē līdzīgu funkcionalitāti elementu izvēlei no saraksta. Galvenā atšķirība ir tā, ka kontrole **ListBox** pieļauj vienlaicīgi izvēlēties vairākus elementus, kā arī saraksts var tikt attēlots vairākās kolonnās.

Kontrole **ListBox** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
ColumnWidth	Atgriež/uzstāda kolonnas platumu, kas tiek izmantots, ja saraksts tiek attēlots vairākās kolonnās.
CustomTabOffsets, UseCustomTabOffsets	Atgriež tabulāciju kolekciju, kuru pēc tam var modificēt, lai pielāgotu saraksta elementu tabulāciju platumus. Ja saraksta elementa nosaukumā ir simbols '\t', tad aiz tā esošais teksts tiek izlīdzināts ar tekošo tabulācijas platumu. Piemēram: <pre> ListBox.IntegerCollection tabStops = listBox1.CustomTabOffsets; tabStops.Add(40); listBox1.UseCustomTabOffsets = true; for (int i = 0; i < 5; i++) listBox1.Items.Add("Elements\t" + i); </pre> 
DrawMode	Atgriež/uzstāda saraksta elementu pārzīmēšanas režīmu. Ja elementi tiek attēloti vairākās kolonnās, tad nav iespējams režīms OwnerDrawVariable ar dažāda izmēra elementu augstumiem.
HorizontalScrollbar	Atgriež/uzstāda horizontālās rītiņas rādīšanu.
IntegralHeight	Atgriež/uzstāda automātisku kontroles augstuma koriģēšanu, lai saraksta elementi netiktu attēloti daļēji.
ItemHeight	Atgriež/uzstāda saraksta elementu augstumu.
Items	Atgriež/uzstāda saraksta elementu kolekciju.
MultiColumn	Atgriež/uzstāda saraksta elementu attēlošanu vairākās kolonnās.
SelectedIndex, SelectedItem	Atgriež/uzstāda izvēlēto saraksta elementu pēc tā indeksa vai objekta. Indeksa rezultāts -1 vai objekta vērtība null nosaka, ka neviens elements nav izvēlēts.
SelectedIndices, SelectedItems	Atgriež/uzstāda izvēlētos saraksta elementu indeksus vai objektus. Lai noskaidrotu visus izvēlētos elementus var izmantot šādu priekšrakstu: <pre> foreach (string elem in listBox1.SelectedItems) MessageBox.Show(elem); </pre>
SelectionMode	Atgriež/uzstāda saraksta elementu izvēles režīmu: <ul style="list-style-type: none"> • None – nav iespējams izvēlēties nevienu saraksta elementu; • One – vienlaicīgi var būt izvēlēts ne vairāk kā viens saraksta elements; • MultiSimple – var būt izvēlēti vairāki elementi, katru izvēli uzstādot vai atceļot ar peles klikšķi vai klaviatūras taustiņu SPACE; • MultiExtended – var būs izvēlēti vairāki elementi, un lietotājs var lietot klaviatūras SHIFT, CTRL un kursora taustiņus, lai veiktu elementu grupu izvēli.
Sorted	Atgriež/uzstāda saraksta elementu attēlošanu sakārtotā secībā.

Text	Atgriež/uzstāda izvēlēto saraksta elementu pēc tā teksta.
TopIndex	Atgriež/uzstāda pirmā redzamā saraksta elementa indeksu.
UseTabStops	Atgriež/uzstāda saraksta elementu nosaukumu sadalīšanu pa tabulācijām.
BeginUpdate(), EndUpdate()	Atslēdz un ieslēdz saraksta elementu pārzīmēšanu – izmanto kad sarakstam ir secīgi jāpievieno vairāki elementi.
ClearSelected()	Atceļ visus izvēlētos elementus.
FindString(), FindStringExact()	Teksta meklēšana saraksta elementos un pirmā atrastā elementa indeksa atgriešana. Atgriež -1, ja meklētais teksts nav atrasts nevienā saraksta elementā.
GetItemHeight()	Atgriež parametrā norādītā indeksa saraksta elementa augstumu.
GetItemRectangle()	Atgriež parametrā norādītā indeksa saraksta elementa augstumu un platumu.
GetSelected()	Atgriež vai parametrā norādītā indeksa saraksta elements ir izvēlēts.
SetSelected()	Uzstāda parametrā norādītā indeksa saraksta elementu kā izvēlētu vai neizvēlētu.

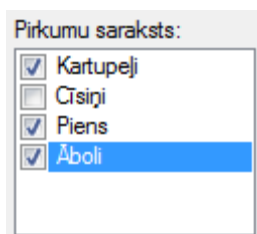
Būtiskākie kontroles **ListBox** notikumi:

SelectedIndexChanged un **SelectedValueChanged**, kas iestājas ja attiecīgi mainās izvēlēta saraksta elementa indekss vai vērtība.

CheckedListBox – izvēles rūtiņu saraksts.

Mantošanas hierarhija: *CheckedListBox* > *ListBox* > *ListControl* > *Control*

Ja izvēles rūtiņu skaits ir mainīgs, vai arī tas ir jāpielieto kādam mainīga skaita identifikatoru sarakstam, tad vairāku kontroļu **CheckBox** vietā lietderīgāk ir izmantot izvēles rūtiņu saraksta kontroli **CheckedListBox**, kura ir atvasināta no saraksta kontroles **ListBox**, piemēram:



Tajā izvēles rūtiņu elementus var uzskaitīt īpašībā **Items**, bet lai noskaidrotu, pašreiz aktīvo izvēles rūtiņu, jāizmanto metode **SelectedItem**, pirms tam pārliccinoties, vai vispār kāds saraksta elements ir aktīvs (kā iepriekšējā attēlā ‘Āboli’):

```
if (checkedListBox1.SelectedIndex >=0)
    label1.Text = checkedListBox1.SelectedItem.ToString();
```

Tā kā sarakstā vienlaicīgi var būt izvēlētas vairākas rūtiņas, tad izvēlēto izvēles rūtiņu kopas iegūšanai jāizmanto īpašība **CheckedItems**. Ja ir svarīgi zināt tikai izvēlēto izvēles rūtiņu kārtas numurus, jāizmanto īpašība **CheckedIndices**.

Jaunu elementa pievienošanu veic ar **Items** metodi **Add(<teksts>, <stāvoklis>)**, esošā nodzēšanu ar **Remove(<teksts>)**, bet visa saraksta dzēšanu – ar metodi **Clear()**.

Kontroles **CheckedListBox** notikumi:

- **SelectedIndexChanged** – izmainīta izvēlēta rūtiņa;
- **SelectedValueChanged** – izvēlētajai rūtiņai izmainījās vērtība.

PictureBox – attēla lodziņš.

Mantošanas hierarhija: *PictureBox* > *Control*

Kontroles galvenais uzdevums ir formās attēlot dažāda formāta attēlus. Kaut gan attēlu var piesaistīt praktiski jebkurai kontrolei, tomēr šī kontrole nodrošina papildus iespējas.

Kontrole **PictureBox** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
BorderStyle	Atgriež/uzstāda kontroles robežu stilu.
ErrorImage	Atgriež/uzstāda attēlu, kas tiek rādīts kontrolē, ja kontroles īpašības Image attēls nevar tikt ielādēts, vai arī tā ielādes process tiek atcelts.
Image	Atgriež/uzstāda attēla instanci, kas tiks rādīta kontrolē.
ImageLocation	Atgriež/uzstāda URL saiti uz attēlu, kurš tiks dinamiski lejupielādēts programmas izpildes laikā un piešķirta īpašībai Image .
InitialImage	Atgriež/uzstāda attēlu, kas tiks rādīts, kamēr attēls tiek lejupielādēts no saites, uz ko norāda īpašības ImageLocation vērtība.
SizeMode	Atgriež/uzstāda attēla mērogošanas režīmu: <ul style="list-style-type: none">• Normal – attēls netiks mērogots. Ja attēls ir lielāks par kontroles izmēriem, tiks attēlota attēla kreisā augšējā mala;• StretchImage – attēls tiks mērogots, lai pilnībā aizņemtu kontroles izmērus;• AutoSize – kontroles izmēri tiks automātiski mainīti, lai atbilstu attēla izmēriem;• CenterImage – attēls tiks centrēts attiecībā pret kontroles izmēriem. Ja attēls ir lielāks par kontroles izmēriem, tiks rādīts tikai attēla centrs;• Zoom – attēls tiks proporcionāli mērogots, lai tā garākā mala pilnībā aizpildītu atbilstošo kontroles malu.
WaitOnLoad	Atgriež/uzstāda sinhronu (true) vai asinhronu (false) attēla ielādes režīmu.
CancelAsync()	Atceļ attēla asinhronas ielādes procesu.
Load(), LoadAsync()	Sinhroni vai asinhroni ielādē attēlu.

Notikumi, uz kurus visbiežāk nākas apstrādāt:

- **LoadProgressChanged** – attēla asinhronas ielādes progresu palielināšanās;
- **LoadCompleted** – attēla asinhrona ielāde beigusies.

ProgressBar – progresu josla.

Mantošanas hierarhija: *ProgressBar* > *Control*

Progresu josla tiek izmantota, lai attēlotu ilgstoši izpildāma procesa progresu – tā vietā, lai lietotājs domātu, ka lietojumprogramma ir ‘uzkārusies’ tiek izvadīta progresējoša josla, kas vizuāli norāda procesa pabeigtības procentu vai procesa norisi.

Kontrole **ProgressBar** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
MarqueeAnimation-	Atgriež/uzstāda markīzes animācijas ātrumu.

Speed	
Maximum	Atgriež/uzstāda progresu maksimālo vērtību. Pēc noklusēšanas 100.
Minimum	Atgriež/uzstāda progresu minimālo vērtību. Pēc noklusēšanas 0.
Step	Atgriež/uzstāda progresu palielināšanas soli, kas tiks izmantots izsaucot metodi PerformStep() .
Style	Atgriež/uzstāda progresu joslas stilu: <ul style="list-style-type: none"> • Blocks – kā klucīši; • Continuous – nepārtraukts; • Marquee – kā animētu markīzi. Izmanto, ja nav jārada progress, bet vizuāla animācija, ka process notiek.
Value	Atgriež/uzstāda progresu vērtību. Tai jābūt diapazonā [Minimum ; Maximum], pretējā gadījumā tiks izraisīts izņēmums ar tipu ArgumentException .
Increment()	Palielina īpašības Value vērtību par argumentā norādīto lielumu.
PerformStep()	Palielina īpašības Value vērtību par īpašības Step vērtību. Šo metodi nevar izsaukt, ja progresu joslas stils ir animēta markīze.

Progresu joslai parasti neveic notikumu apstrādi, tomēr vairāki notikumi tai ir pieejami.

Panel – panelis.

Mantošanas hierarhija: *Panel > ScrollableControl > Control*

Panela vienīgais uzdevums ir būt par konteineri citām kontrolēm. Tā kā panelis ir klases **ScrollableControl** apakšklase, tad tas automātiski nodrošina arī ritināšanas funkcionalitāti.

GroupBox – grupēšanas lodziņš.

Mantošanas hierarhija: *GroupBox > Control*

Kontrolei **GroupBox**, līdzīgi kā kontrolei **Panel** ir tikai viens uzdevums – būt par konteineri citām kontrolēm. Grupēšanas lodziņš papildus piedāvā izvadīt grupas nosaukumu, kā arī, ja nepieciešams, attēlot rāmīti.

Visbiežāk šo kontroli izmanto par konteineri kontroļu **RadioButton** (radiopoga) izvietojumam, tādā veidā ļaujot uz vienas formas uzvietot vairākas radiopogu grupas, katrā no kurām vienlaicīgi var būt aktivizēta tikai viena radiopoga.

TabControl – cilnes kontrole.

Mantošanas hierarhija: *TabControl > Control*

Cilnes kontrole pieder pie konteineru komponentu grupas, kuras uzdevums ir nodrošināt citu kontroļu izvietojumu savu izmēru robežās. Kontrole **TabControl** sevī ļauj izvietot tikai kontroles **TabPage** (kura, savukārt, ir kontroles **Panel** apakšklase), kurās tiek izvietotas kontroles. Kontrole **TabControl** nodrošina, ka vienlaicīgi var redzēt tikai vienu kontroles **TabPage** instanci – uz pārējām var pārslēgties tikai uzklikšķinot uz to cilņu nosaukumiem, kurus definē ar kontroles **TabPage** īpašību **Text**.

Kontrole **TabControl** definē šādas īpašības un metodes:

Īpašība/Metode()	Apraksts
Alignment	Atgriež/uzstāda cilnes nosaukumu attēlošanas vietu – augšā, apakšā, pa kreisi vai

	pa labi.
Appearance	Atgriež/uzstāda cilnes nosaukumu attēlošanas stilu.
Multiline	Atgriež/uzstāda cilnes nosaukumu rādīšanu vairākās rindās, ja cilnes neievietoja vienā rindā.
SelectedIndex	Atgriež/uzstāda aktīvās cilnes indeksu.
SelectedTab	Atgriež/uzstāda aktīvo cilni.
TabPage	Atgriež tekošo cilņu TabPage instanču kolekciju.
SelectTab()	Aktivizē parametrā norādīto cilni.

ErrorProvider – kļūdu apstrādātājs.

Mantošanas hierarhija: *ErrorProvider* > *Component*

Kā redzams no mantošanas hierarhijas, tad šis ir nevizuāls komponents (jo tā priekštečos nav klase **Control**), kas atrodama komponentu paletes kategorijā **Components**. Šī komponenta mērķis ir vienkāršot kļūdu apstrādi formas vizuālajām kontrolēm. Tas nodrošina labāku alternatīvu nekā ziņojumu loga izvade, jo realizē vizuālu izcēlumu pie kontroles, kuras vērtība ir nekorekta, ļaujot to vizuāli izcelt visu laiku, kamēr kļūda nav novērsta.

Šo komponentu visbiežāk izmanto šādā scenārijā:

1. Kādas kontroles notikuma **Validating** apstrādātājā veic kontrolē ievadītās vērtības pārbaudi, un, ja tiek konstatēta kļūda, izsauc kļūdu apstrādātāja instances metodi **SetError()**, kuras parametrus norāda to kontroli un kļūdu paskaidrojošu tekstu, kurai ir nekorekta vērtība, piem.:

```
errorProvider.SetError(txtDefaultFolder
    , "Katalogam ir jābūt izveidotam un pieejamam.");
```

2. Ja kontroles vērtības pārbaude ir veiksmīga, tad kļūdas situācija ir jāatceļ, izsaucot šo pašu metodi, tikai kļūdas teksta vietā norādot tukšu virkni, piem., notikuma **Validated** apstrādātājā rakstot:

```
errorProvider.SetError(txtDefaultFolder, "");
```

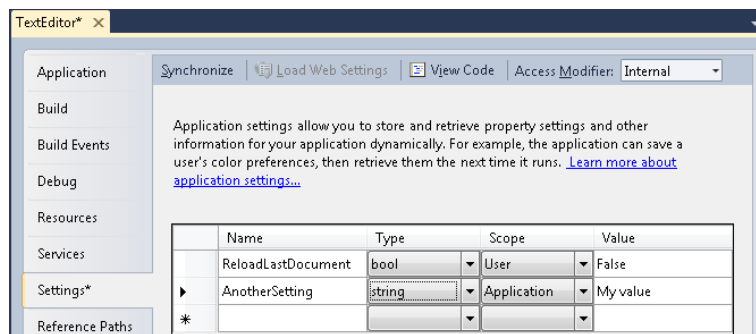
Komponentu un kontroļu paletes logā ir pieejamas vēl daudzas citas kontroles un komponenti, bet visiem ir līdzīgi principi to izmantošanai, tāpēc, ja izstrādātājs pārzina galveno kontroļu izmantošanas principus, tad jaunu vai līdzīgu kontroļu un komponentu apguve parasti nerada grūtības.

3.2 LIETOJUMPROGRAMMAS UZSTĀDĪJUMU PĀRVALDĪBA

Jebkurai lietojumprogrammai tiek izvirzītas prasības būt pielāgojamai konkrēta lietotāja vajadzībām vai uzņēmuma politikai, tāpēc pilnvērtīga lietojumprogramma nevar iztikt bez uzstādījumu (opciju) dialoga, kā arī uzstādījumu datu saglabāšanai starp lietojumprogrammas palaišanas reizēm.

.NET lietojumprogrammās ir pieejams lietojumprogrammas uzstādījumu (**Application Settings**) norādīšanas, saglabāšanas un labošanas mehānisms, kas vienlaicīgi arī atvieglo uzstādījumu piesaisti komponentiem.

Uzstādījumu vizuāla definēšana notiek projekta īpašību loga cilnē **Settings** (projekta īpašību logu atver risinājuma pārlūka logā no virsotnes **Properties** konteksta izvēlnes izvēloties punktu **Properties**):




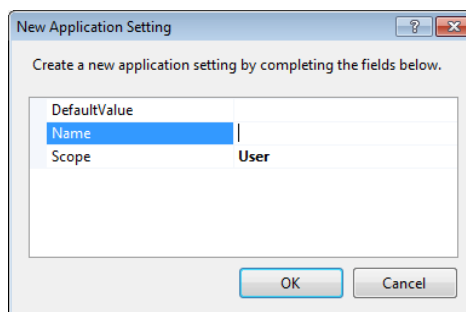
Lietojumprogrammas uzstādījumu raksturo:

- Tā nosaukums (**Name**), kuram jāatbilst C# identifikatora prasībām, un jābūt unikālam lietojumprogrammas ietvaros;
- Tips (**Type**), kas nosaka, kāda veida informācija uzstādījums glabā. Tiek piedāvāts uzstādījumus saglabāt visos iebūvētajos tipos, kā arī klašu tipiem (piem., **Font**, **Color**, **Size**);
- Darbības apgabals (**Scope**) – ir pieejami divu veidu apgabali. Lietojumprogrammas apgabala (**Application Scope**) uzstādījumus lietotājs var izmantot tikai lasīšanas režīmā, un tie ir kopēji visiem lietotājiem. Lietotāja apgabala (**User Scope**) uzstādījumus var mainīt katrs lietojumprogrammas lietotājs, un katram lietotājam tiek saglabāta sava uzstādījumu kopija, kuru maiņa neietekmē pārējo lietotāju lietotāja apgabala uzstādījumus;
- Vērtība (**Value**) – uzstādījuma tekošā vērtība, kura atbilst tā tipam.

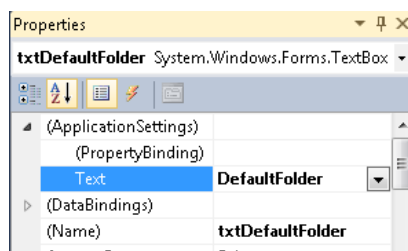
Lietotāja uzstādījumu izmantošanas priekšrocība ir tā, ka tos var piesaistīt konkrētai formas kontrolei – formu ielādējot un atverot lietojumprogrammas izpildes laikā, kontrolē tiks ielādēta uzstādījuma tekošā vērtība, bet aizverot – to var automātiski saglabāt.

Lai piesaistītu uzstādījumu formas kontrolei, jārīkojas šādi:

1. Formas redaktorā aktivē nepieciešamo kontroli un īpašību redaktora logā izvērs īpašību „**(Application Settings)**”, zem kuras uzklikšķina uz īpašības „**(PropertyBindings)**” pogas .
2. Dialogā, kas atveras ir iespēja piesaistīt kontroles noteikta īpašības jau esošam uzstādījuma nosaukumam, vai arī definēt jaunu uzstādījumu:



3. Jauna uzstādījuma dialogā jānorāda uzstādījuma nosaukums (**Name**), darbības apgabals (**Scope**) un noklusētā vērtība (**Default Value**), kas tiks izmantota, ja lietotājs nebūs norādījis savu. Pēc pievienošanas kontroles īpašību logā zem īpašības (**Application Settings**) tiek ievietota jauna rinda, kurā norādīts kontroles īpašības nosaukums un tam piesaistītais uzstādījuma nosaukums:



Piekluve uzstādījumiem no pirmkoda tiek nodrošināta ar automātiski izveidotas C# klases **Settings** palīdzību.

Lai lietotāja uzstādījumi saglabātos starp lietojumprogrammas palaišanas reizēm, pirmkodā ir obligāti jāizsauc uzstādījumu klases metode **Save()**, parasti to veic formas aizvēršanas notikuma **FormClosing** vai apstiprināšanas pogas nospiešanas notikuma apstrādātājos, piemēram:

```
Properties.Settings.Default.Save();
```

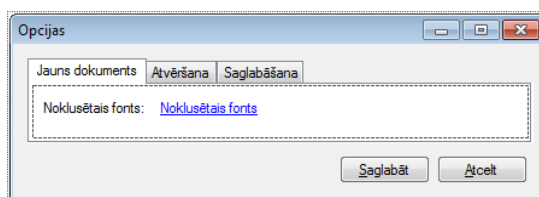
Bet specifiska uzstādījuma nolasīšanai/uzstādīšanai jāizmanto pirmkods šādā formā:

```
Properties.Settings.Default.UzstādījumaNosaukums;
```

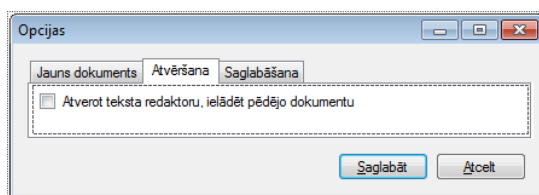
3.3 UZDEVUMS PASNIEDZĒJA VADĪBĀ

Papildināt iepriekšējā laboratorijas darbā izstrādāto teksta redaktoru ar šādu funkcionalitāti.

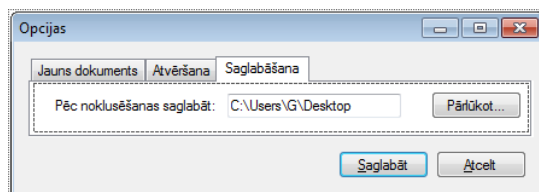
Izveidot uzstādījumu dialogu, kurā būtu izvietota cilnes kontrole ar trīs cilnēm. Pirmajā cilnē „Jauns dokuments” jābūt izvietotām iezīmes un iezīmes ar saiti kontrolēm, uz kuras uzklikšķinot tiktu atvērts fonta izvēles dialogs un pēc tā aizvēršanas saites nosaukumā tiktu izvadīts izvēlētais fonta nosaukums un izmērs:



Otrajā cilnē „Atvēršana” jābūt izvēles rūtiņai ar nosaukumu „Atverot teksta redaktoru, ielādēt pēdējo dokumentu”:



Trešajā cilnē „Saglabāšana” jābūt teksta elementam ar iezīmi „Pēc noklusēšanas saglabāt:”, kurā lietotājam ir iespēja norādīt noklusēto katalogu, kurā tiktu piedāvāts saglabāt jaunus dokumentus pēc noklusēšanas, un pogai „Pārlūkot...”, kuru nospiežot, lietotājam tiktu piedāvāts no dialoga izvēlēties noklusēto katalogu:



Šim nolūkam, izveidot kopiju 2. laboratorijas darba projekta katalogam 'Lab2' un nosaukt to par 'Lab3'. Izstrādes vidē atvērt šajā katalogā esošo risinājumu (Lab2.sln), un projekta pārlūkā pārsaukt risinājuma nosaukumu no 'Lab2' uz 'Lab3'.

Risinājuma **Lab3** projektam **TextEditor** pievienot jaunu formu (piemēram, no risinājuma pārlūka loga uz virsotnes **TextEditor** konteksta izvēlnes izvēlēties punktu **Add | Windows Form...**), tās failu nosaukt par **frmOptions.cs**, un uzstādīt īpašības:

Īpašība:	Vērtība:
FormBorderStyle	FixedDialog
MaximizeBox	False
MinimizeBox	False
ShowInTaskbar	False
Size	445; 155
StartPosition	CenterParent
Text	Opcijas

Uz jaunizveidotās formas **frmOptions** uzvietot divas **Button** kontroles, un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	btnSave
Anchor	Bottom, Right
DialogResult	OK
Location	270; 90
Text	&Saglabāt

Īpašība:	Vērtība:
(Name)	btnCancel
Anchor	Bottom, Right
DialogResult	Cancel
Location	350; 90
Text	At&celt

Formas **frmOptions** redaktorā aktivēt pašu formu un uzstādīt īpašības (ieteicams vērtības izvēloties no kombinētā saraksta):

Īpašība:	Vērtība:
AcceptButton	btnSave
CancelButton	btnCancel

No kontroļu kategorijas **Containers**, uzvietot uz formas **frmOptions** cilnes kontroli **TabControl**, un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	tbcOptions
Anchor	Top, Bottom, Left, Right
Location	12; 12
Size	420; 70

Cilnes kontroles instancē **tbcOptions**, izvēlēties cilni **tabPage1** un pēc tam uzklikšķināt uz tās laukuma (lai aktivizētu cilnes īpašības), un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	tbpNewDocument
Text	Jauns dokuments

Kamēr ir aktīva cilne **tbpNewDocument**, uzvietot uz tās iezīmes kontroli **Label**, un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	lblDefaultFont
Location	12; 12
Text	Noklusētais fonts:

Kamēr ir aktīva cilne **tbpNewDocument**, uzvietot uz tās kontroli **LinkLabel**, un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	lnkDefaultFont
Location	105; 12
Text	Noklusētais fonts


Formas redaktorā izvēlēties cilni **tabPage2**, aktivizēt cilnes īpašības, un uzstādīt īpašības:

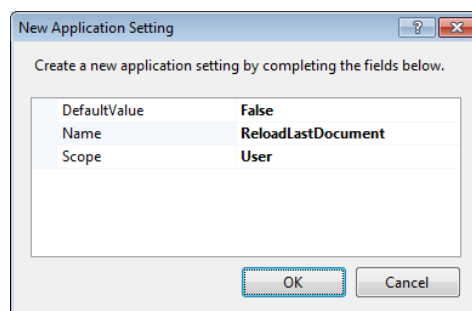
Īpašība:	Vērtība:
(Name)	tbpOpen
Text	Atvēršana

Kamēr ir aktīva cilne **tbpOpen**, uzvietot uz tās izvēles rūtiņas kontroli **CheckBox**, un uzstādīt īpašības:

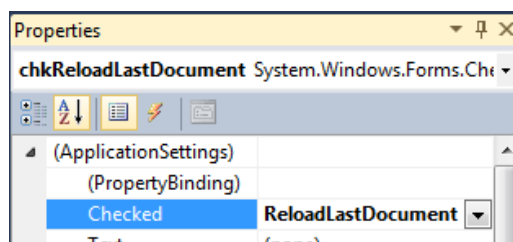
Īpašība:	Vērtība:
(Name)	chkReloadLastDocument
Location	12; 12
Text	Atverot teksta redaktoru, ielādēt pēdējo dokumentu

Lai kontroles **chkReloadLastDocument** īpašībai **Checked** piesaistītu lietojumprogrammas uzstādījumu, rīkoties šādi.

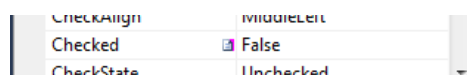
Kontroles **chkReloadLastDocument** īpašību logā izvēlēties īpašības „(Application Settings)” apakšpunktu (Property Binding) un nospiegt . Dialogā, kas atvēršies, izvēlēties īpašību **Checked** un, tās vērtības laukā, izvērst kombinēto sarakstu, un izvēlēties punktu **New...** Dialogā, kas atvēršies, ievadīt šādas vērtības:



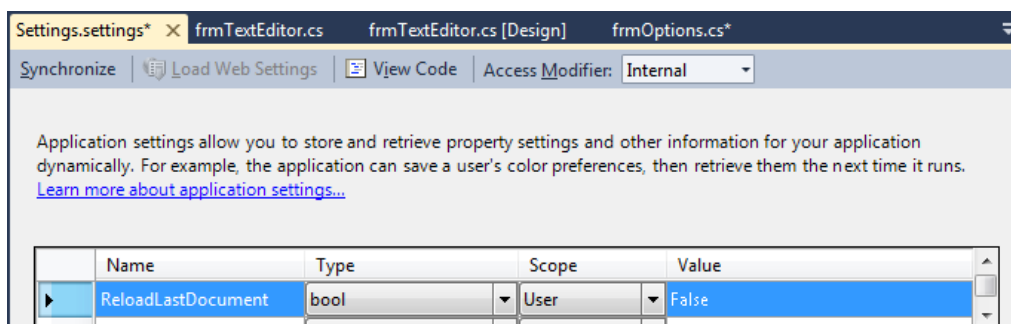
Nospiežot pogu OK divas reizes, aizvērt abus dialogus un pārliecināties, ka īpašībai **Checked** ir piesaistīts lietojumprogrammas uzstādījums **ReloadLastDocument**:



Arī pie pašas īpašības nosaukuma tiek parādīta ikona, kas norāda uz to, ka šīs īpašības vērtība ir sasaistīta ar lietojumprogrammas uzstādījumu:



Ar peles dubultklikšķi uz risinājuma pārlūka virsotnes **TextEditor | Properties | Settings.settings**, atvērt lietojumprogrammas uzstādījumu konfigurācijas logu, un ievērot, ka tajā parādās izveidotais uzstādījums:



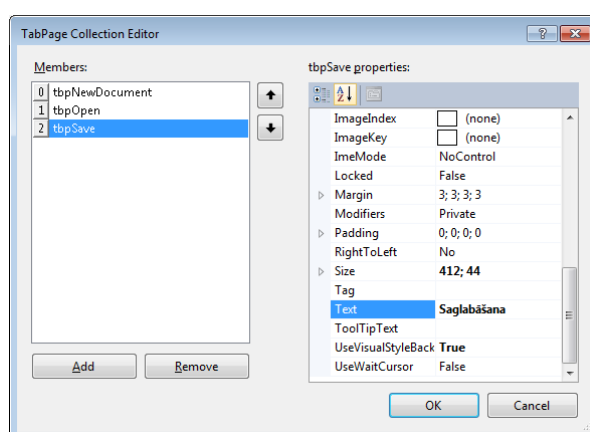
Šajā logā pievienot divus jaunus uzstādījumus ar šādām īpašībām:

Name	Type	Scope	Value
DefaultFolder	string	User	
DefaultFont	System.Drawing.Font	User	Microsoft Sans Serif; 8,25pt

Aizvērt lietojumprogrammas uzstādījumu logu, saglabājot veiktās izmaiņas.

Formas **frmOptions** redaktorā izvēlēties kontroli **tbOptions** un atvērt tās īpašības **TabPage** dialogu, kurā ar pogu **Add** pievienot vēl vienu cilni un dialoga labajā pusē uzstādīt šādas īpašības:

Īpašība:	Vērtība:
(Name)	tbpSave
Text	Saglabāšana



Nospiežot pogu OK, aizvērt šo dialogu un formas redaktorā aktivizēt jaunpievienoto cilni **tbpSave**.

Cilnē **tbpSave** uzvietot iezīmes kontroli **Label** un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	lblDefaultFolder
Location	12; 12
Text	Pēc noklusēšanas saglabāt:

Cilnē **tbpSave** uzvietot kontroli **TextBox** un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	txtDefaultFolder
Location	158; 9
Size	144; 20

Cilnē **tbpSave** uzvietot kontroli **Button** un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	btnBrowse
Location	330; 7
Text	Pārlūkot...

Formas **frmOptions** redaktorā uzvietot šādus nevizuālos komponentus, un uzstādīt to nosaukumus:

Komponents:	Nosaukums (Name):
FontDialog	dlgFont
FolderBrowserDialog	dlgBrowser
ErrorProvider	erpOptions

Formas redaktorā izvēlēties pašu formu **frmOptions** un pievienot notikuma **Load** apstrādātāju ar šādu realizāciju:

```
private void frmOptions_Load(object sender, EventArgs e)
{
    // no uzstādījumiem nolasa lietojumprogrammas uzstādījumu
    fontSetting = Properties.Settings.Default.DefaultFont;
    txtDefaultFolder.Text = Properties.Settings.Default.DefaultFolder;
    // aktualizē iezīmi ar saiti par noklusēto fontu:
    RefreshDefaultFontLinkLabel();
}

private Font fontSetting;

private void RefreshDefaultFontLinkLabel()
{
    if (fontSetting != null)
    {
        // iezīmei ar saiti nomainīt rādāmo fontu un tā izmēru
        lnkDefaultFont.Text = string.Format("{0} ({1})",
            fontSetting.Name, fontSetting.Size);
    }
}
```

Cilnē **tbpNewDocument** izvēlēties kontroli **lnkDefaultFont** un tai izveidot jaunu notikuma **LinkClicked** apstrādātāju, ar šādu realizāciju (ievērot, ka klasē **frmOptions** ir jāddefinē jauna metode **RefreshDefaultFontLinkLabel()**):

```
private void lnkDefaultFont_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    // inicializē un parāda fonta izvēles dialogu
    dlgFont.Font = fontSetting;
    if (DialogResult.OK == dlgFont.ShowDialog())
    {
        // saglabā dialogā izvēlēto fontu
        fontSetting = dlgFont.Font;
        RefreshDefaultFontLinkLabel();
    }
}
```

Aktivizēt cilni **tbpSave**, izvēlēties kontroli **txtDefaultFolder**, un tai piesaistīt notikuma **Validating** apstrādātāju ar šādu realizāciju:

```
private void txtDefaultFolder_Validating(object sender
, CancelEventArgs e)
{
    // nosaka, vai izvēlētais katalogs eksistē
    e.Cancel = !System.IO.Directory.Exists(txtDefaultFolder.Text);
    if (e.Cancel)
        // ja katalogs neeksistē, uzstādīt teksta lauka kļūdu
        erpOptions.SetError(txtDefaultFolder
, "Katalogam ir jābūt izveidotam un pieejamam.");
}
```

Kontrolei **txtDefaultFolder** pievienot notikuma **Validated** apstrādātāju (šis notikums izpildīsies tikai tad, ja notikuma **Validating** parametra **e** īpašība **Cancel** netiks uzstādīta uz **true**):

```
private void txtDefaultFolder_Validated(object sender, EventArgs e)
{
    // atceļ lauka kļūdu, ja tā bija uzstādīta
    erpOptions.SetError(txtDefaultFolder, "");
}
```

Cilnē **tbpSave** un izvēlēties kontroli **btnBrowse**, un izveidot notikuma **Click** apstrādātāju:

```
private void btnBrowse_Click(object sender, EventArgs e)
{
    // uzstādīt katalogu pārlūka dialoga parametrus
    dlgBrowser.SelectedPath = txtDefaultFolder.Text;
    dlgBrowser.ShowNewFolderButton = true;

    if (DialogResult.OK == dlgBrowser.ShowDialog())
        // teksta laukā saglabāt izvēlēto katalogu
        txtDefaultFolder.Text = dlgBrowser.SelectedPath;
}
```

Formas **frmOptions** kontrolei **btnSave** pievienot notikuma **Click** apstrādātāju ar šādu realizāciju:

```
private void btnSave_Click(object sender, EventArgs e)
{
    //uzstādījumu apstiprināšana
    Properties.Settings.Default.DefaultFont = fontSetting;
    Properties.Settings.Default.DefaultFolder = txtDefaultFolder.Text;
    //uzstādījumu vērtību saglabāšana
    Properties.Settings.Default.Save();
}
```

Nokompilēt risinājumu un pārlicināties, ka tajā nav kļūdas.

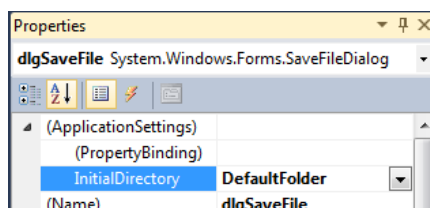
Formas **frmTextEditor** galvenās izvēlnes kontrolei **mnuMainMenu**, pievienot jaunu apakšpunktu izvēlei „Rediģēt”, un uzstādīt īpašības:

Īpašība:	Vērtība:
(Name)	mnuOptions
Text	&Opcijas...

Šim izvēlnes punktam pievienot notikuma **Click** apstrādātāju:

```
private void mnuOptions_Click(object sender, EventArgs e)
{
    frmOptions options = new frmOptions();
    if (DialogResult.OK == options.ShowDialog())
        txtContent.Font = Properties.Settings.Default.DefaultFont;
}
```

Atvērt formas **frmTextEditor** redaktoru un no formas nevizuālo komponentu laukuma izvēlēties kontroli **dlgSaveFile**. Izmantojot tādu pašu soļu secību, kā kontrolei **chkReloadLastDocument**, kontroles **dlgSaveFile** īpašībai **InitialDirectory** piesaistīt lietotjumprogrammas uzstādījumu **DefaultFolder** (ievērot, ka tas jau ir izveidots iepriekš, tāpēc atliek tikai izvēlēties no piedāvātā saraksta):



Uzlabot teksta redaktora funkcionalitāti, lai attiecīgie izvēlnes punkti un rīku joslas ikonas tiktu automātiski ieslēgtas (Enabled), vai atslēgtas (Disabled), atkarībā no tā, vai atbilstošā komanda dotajā brīdī ir pieejama izpildei vai nav. Šim nolūkam formas **frmTextEditor** klasē definēt jaunu metodi – notikuma **Idle** apstrādātāju:

```
void Idle(object sender, EventArgs e)
{
    // katrai iesaistītajai kontrolei uzstādām īpašību Enabled uz
    // true vai false, atkarībā no tā vai tā dotajā brīdī ir izpildāma
    mnuCopy.Enabled = txtContent.SelectionLength > 0;
    mnuCut.Enabled = mnuCopy.Enabled;
    mnuPaste.Enabled =
        Clipboard.GetDataObject().GetDataPresent(DataFormats.Text) == true;
    cutToolStripButton.Enabled = mnuCopy.Enabled;
    copyToolStripButton.Enabled = mnuCut.Enabled;
    pasteToolStripButton.Enabled = mnuPaste.Enabled;
}
```

Lai notikuma **Idle** apstrādātāju piesaistītu lietojumprogrammas dīkstāves notikumam, definēt formas **frmTextEditor** notikuma **Load** apstrādātāju ar šādu realizāciju:

```
private void frmTextEditor_Load(object sender, EventArgs e)
{
    Application.Idle += new EventHandler(Idle);
    // nolasa lietojumprogrammas uzstādījumu noklusētajam fontam
    txtContent.Font = Properties.Settings.Default.DefaultFont;
}
```

Nokompilēt lietojumprogrammu, palaist to uz izpildi un pārlicināties, ka:

- Strādā uzstādījumu dialoga atvēršana no izvēlnes;
- Uzstādījumu dialogs ir modāls – kamēr tas ir atvērts, teksta redaktora logam piekļūt nevar;
- Strādā uzstādījumu mainīšanas iespēja – noklusētā fonta maiņa un noklusētā kataloga maiņa, kā arī šo uzstādījumu saglabāšana starp teksta redaktora palaišanām uz izpildi;
- Noklusētā kataloga esamības pārbaude un kļūdas nodrošinātāja ikonas un teksta atspoguļošana nekorekta kataloga norādīšanas gadījumā;
- Rīku joslas un izvēlnes punkti „Kopēt”, „Izgriezt” un „Ievietot” ir aktīvi tikai tad, ja šīs komandas var izpildīt.