

Colegiul Național “Liviu Rebreanu” Bistrița

LUCRARE DE ATESTAT

Animarea operațiilor cu vectorii

Absolvent: Profesor coordonator:

Șofran Daniel Mureșan Vasile Ciprian

Bistrița 2021

Cuprins

1. Descriere generală
2. Modalitate de implementare
3. Clasa VectorControl
4. Codul sursă
5. Manual de utilizare
6. Bibliografie
7. Descriere generală

Aplicația **Vectori** este destinată elevilor de clasa a IX-a care studiază capitolul Vectori, profesorilor care predau la clasa a IX-a sau programatorilor începători care doresc să înțeleagă tablourile unidimensionale sau vectorii într-un mod grafic. Aplicația conține teoria referitoare la operațiile pe vectori, cod sursă în c++ și animarea acestor operații, dar și un editor de vectori, care ajută la rezolvarea problemelor cu secvențe.

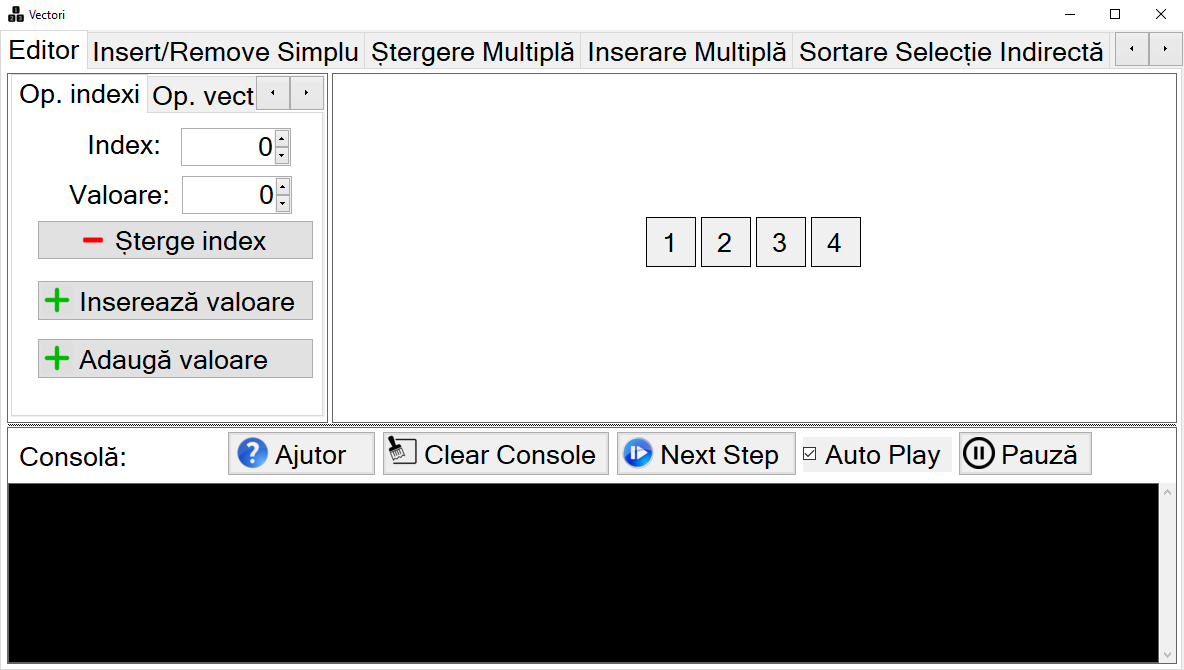
Aplicația este organizată în 9 pagini, denumite astfel:

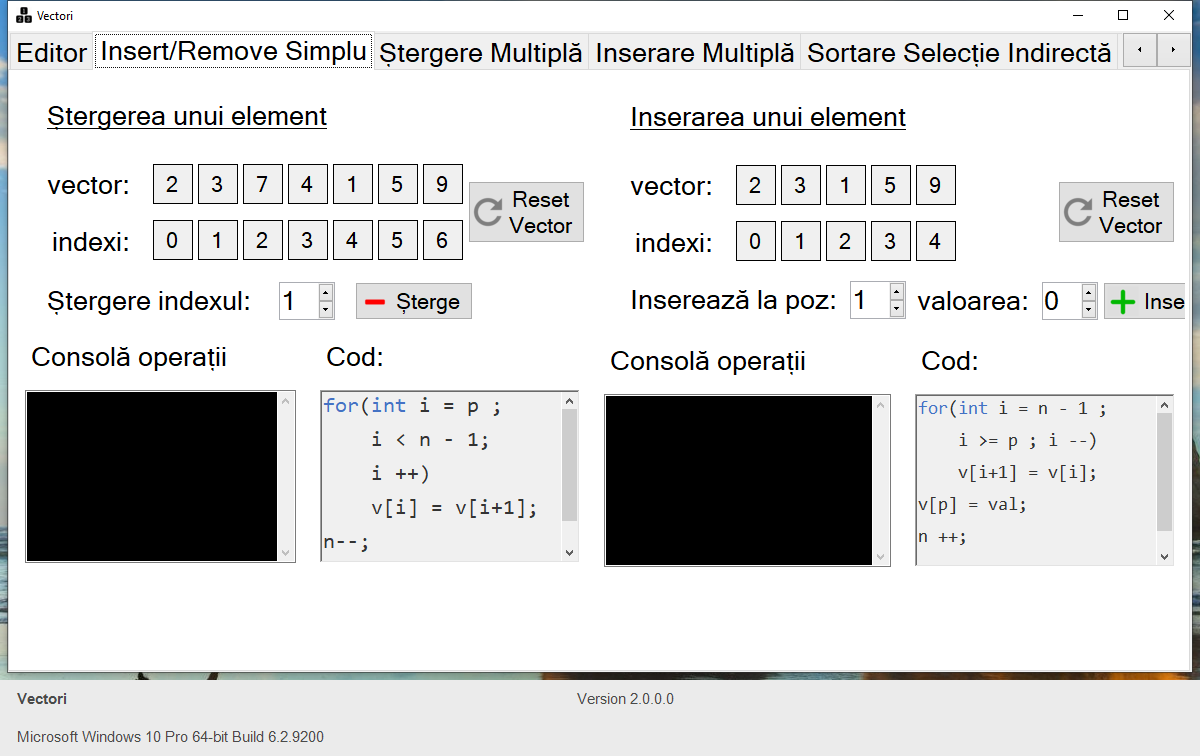
* Editor : Pagina ce conține editorul de vectori. Această pagină conține și unele setări generale ale tuturor vectorilor din program.
* Insert/Remove Simplu : prezintă ștergerea sau inserarea unui element în vector
* Ștergere multiplă : prezintă algoritmul corect și explică în mod grafic incorectitudinea algoritnului naiv
* Inserare multiplă : în mod analog ștergerii multiple
* Sortare Selecție Indirectă
* Sortare Selecție Directă
* Sortare Metoda Bulelor
* Sortare Inserție
* Ajutor, conținând un manual de utilizare

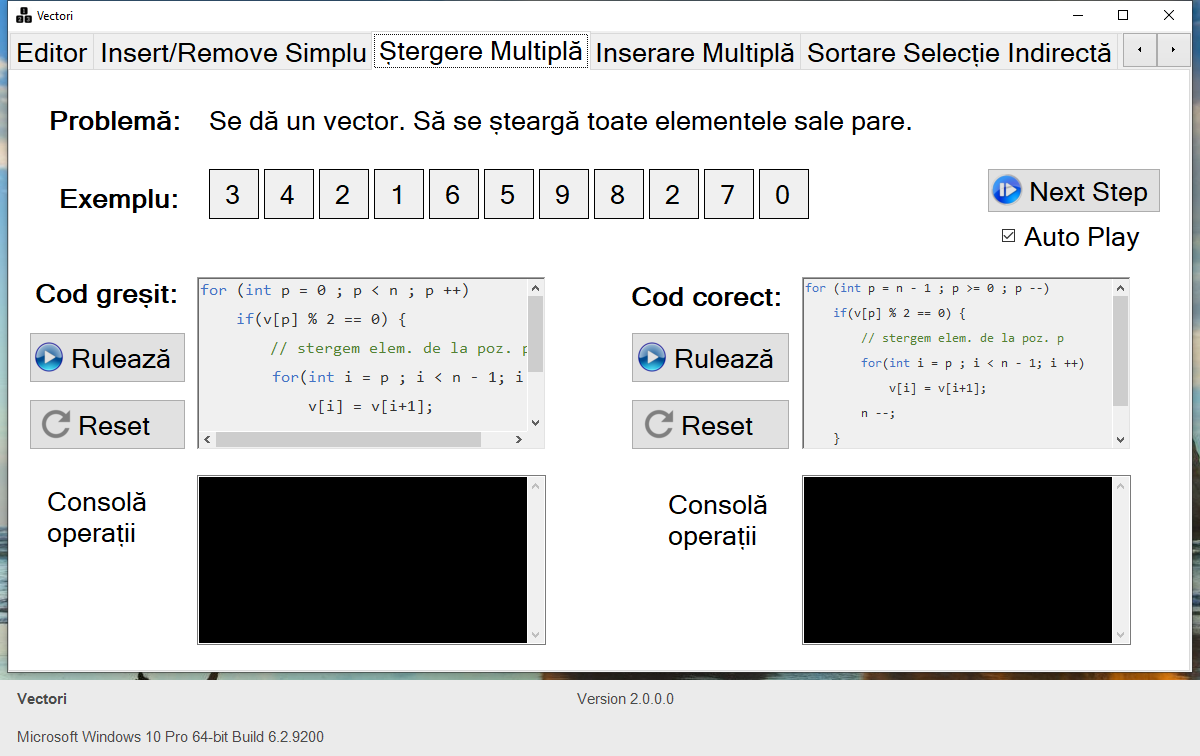
Astfel, fiecare algoritm conține în prezentarea sa :

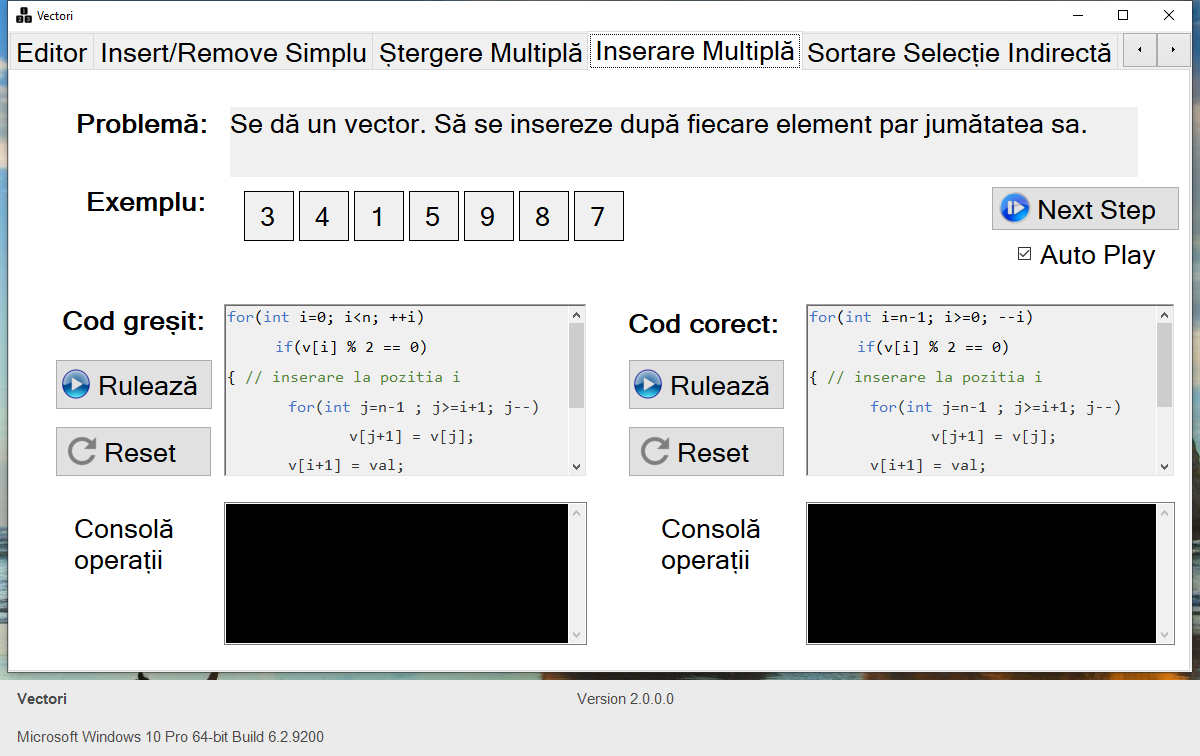
* Descrierea în limbaj natural al algoritmului prezentat
* Codul sursă c++, care stă la baza animației
* Vectorul reprezentat grafic, editabil ca aspect și valori
* Consolă de operații, în care se înregistrează operațiile efectuate

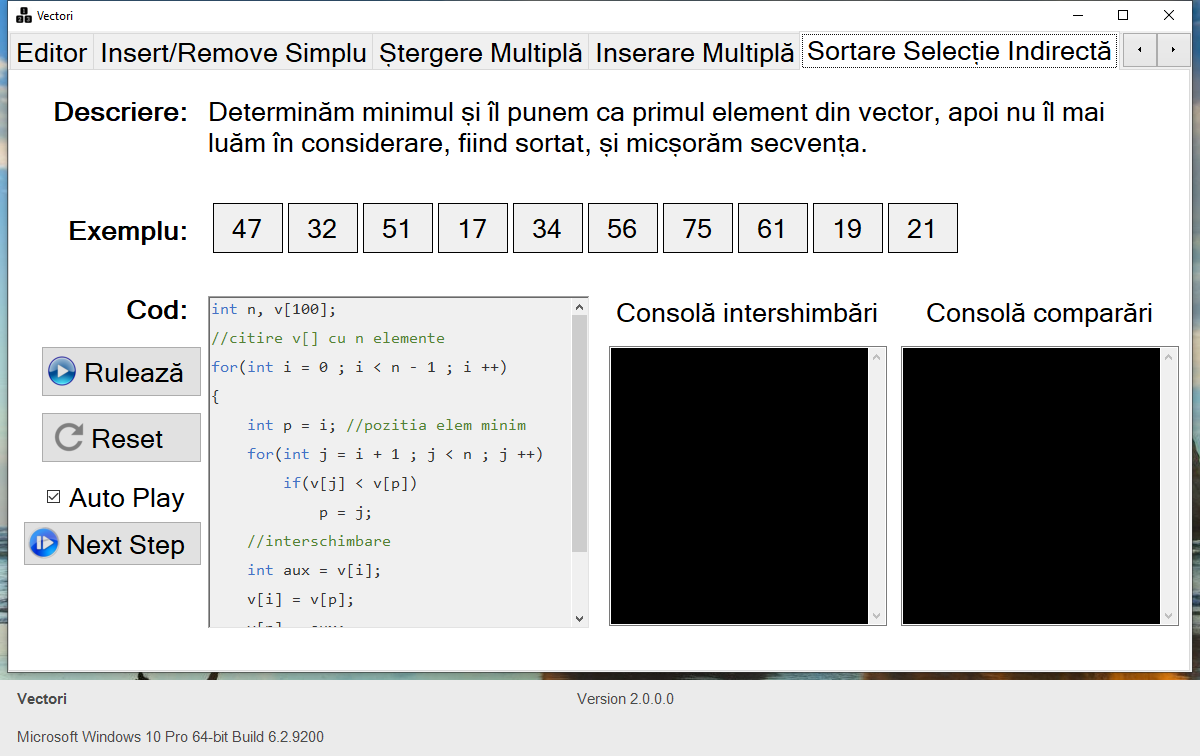
(\*) Prin setări generale vom înțelege schimbarea culorilor de sistem, care merchează o selectare/selecție, o inserare, o ștergere sau o marcare a unei secvențe. O altă setare generală este spațiatorul folosit pentru citirea/scrierea vectorilor din fișier, precum și folosirea sau nu a numărului de elemente, notat în mod normal *n*.

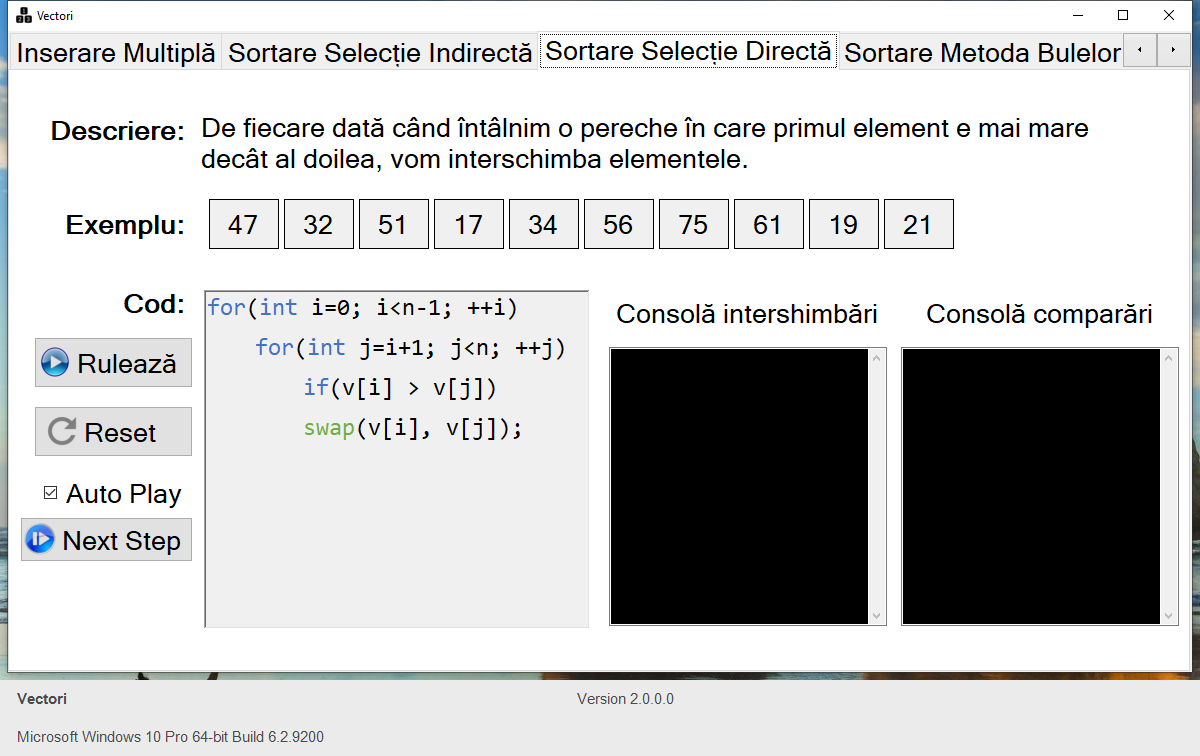


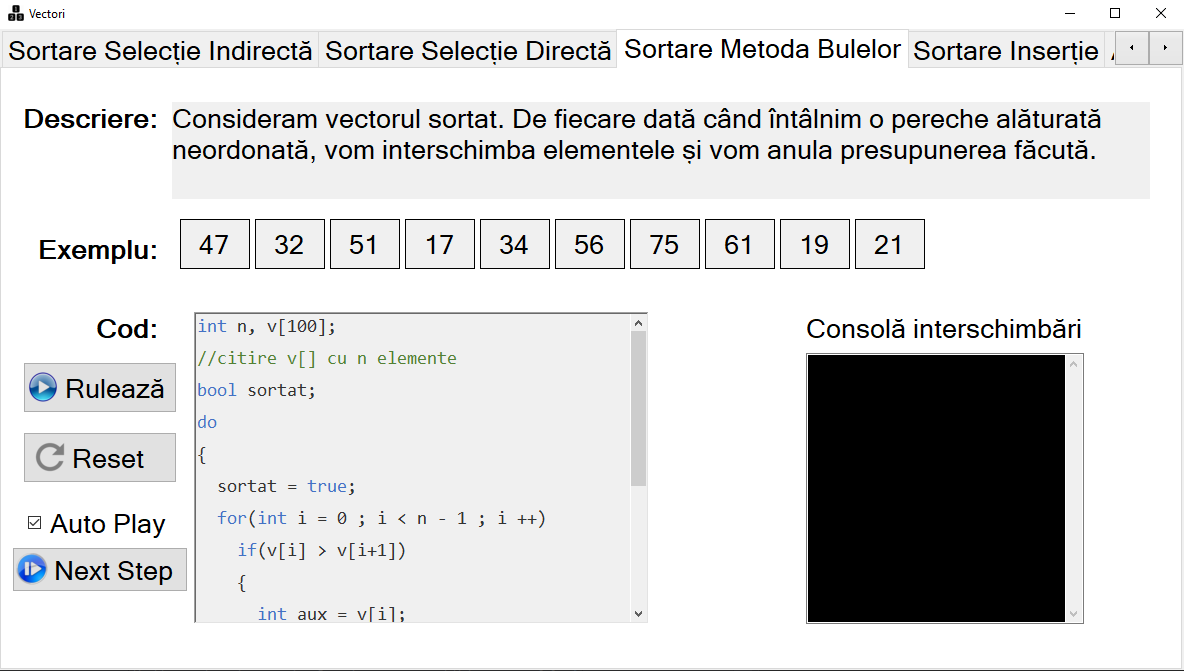


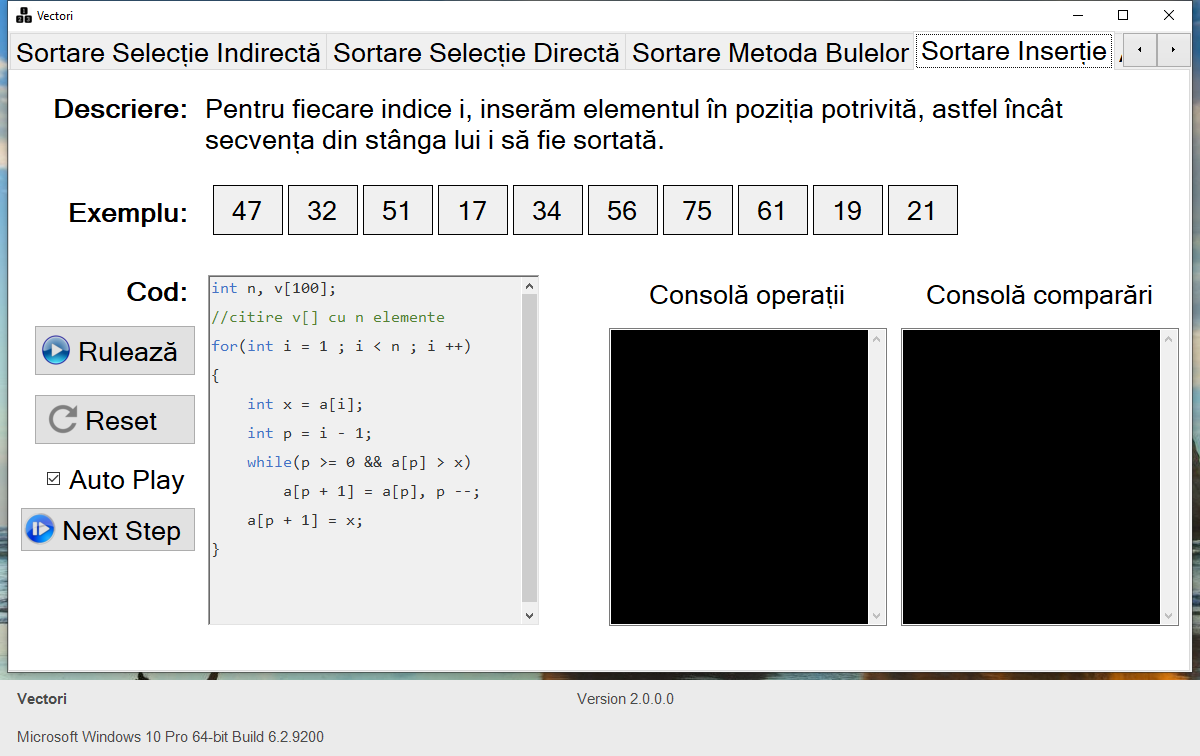








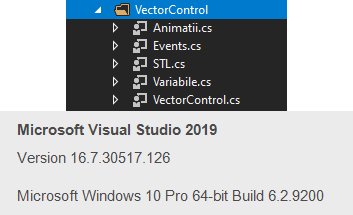


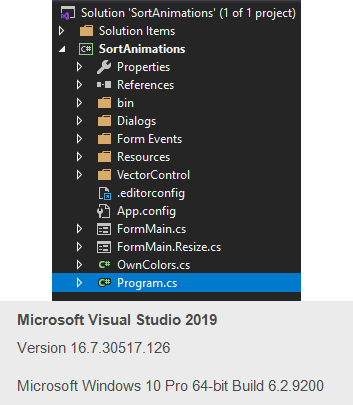


1. Modalitate de implementare

Datorită faptului că nu sunt implementate funcții care animează controale, singura soluție a fost modificarea unor proprietați precum locația sau mărimea controlului treptat, într-un timer, dezavantajul fiind că mărimea controlului este memorată pe baza tipului de date întreg nu zecimal, ceea ce conduce la probleme grele referitoare la schimbarea vitezei animației. Pentru a evita un timer global cu multe if-uri și pentru a favoriza creerea mai ușoară a noi animații, s-a preferat folosirea timer-elor locale. Astfel, fiecare funcție de animare ale clasei VectorControl are un alt timer. Problema apare atunci când, în cod, se apelează de două ori o funcție, în mod succesiv. Astfel, rezultatul este efectuarea simultană a operațiilor și nu succesivă, așa cum este necesar.

Rezolvarea problemei constă în utilizarea unei cozi de asemenea funcții, cărora le adăugăm în ultimul pas al timer-ului apelul unei funcții *ProceedNextInQueue()*, care apelează următoarea funcție din coadă. Astfel, funcțiile inițiale devin un simplu prototip, primind sufixul *\_One*, fiind funcții private din fișierul Animatii.cs din folder-ul VectorControl, fiind necesară definirea a noi funcții care aleg dacă să apeleze efectiv funcția sau să o pună în coadă, funcții aflate în fișierul STL.cs aflate în același folder, numele provenind din faptul că numele funcțiilor este similar cu cel din arhi-cunoscuta librărie c++.





Folderele VectorControl, respectiv SortAnimations

Folderul VectorContr

1. Clasa VectorControl

Pentru a simplifica scrierea codului, a fost creeat un control numit **VectorControl**, care pe langă faptul că permite vizualizarea grafică a unui vector, are implementate funcții de animare a operațiilor pe aceștia, aflate în fișierul *STL.cs* din folderul VectorControl:

* void Swap(int i, int j) – interschimbă cele 2 elemente aflate la indicii *i* și *j*
* void Erase(int i) – șterge din vector elementul de la poziția *i*
* void Insert(int i, int val, bool txt = true) – inserează în vector la poziția *i* valoarea *val*
* void Pop\_back() – șterge ultimul element, dacă există
* void Pop\_front() – șterge primul element, dacă există
* void Add(int val, bool txt = true) - adaugă în vector valoarea *val*
* void Select(params int[] indexes) – selectează indicii dați, colorându-i pentru scurt timp
* void Sleep(int milisecunde) – face o pauză în animație
* void AtTimeAction(Action action, bool manual\_stop = false) – permite definirea unei acțiuni/seturi de operații asupra vectorului, care se executa in momentul cand celelalte animatii s-au finalizat
* void AppendText(string log) – scrie text în consola asociată
* void ShowSwaps() – afișează în consolă numerul de interschimbări efectuate în vectorul respectiv
* void ColorDefault() – resetează culorile de fundal ale elementelor

, *txt* determinând dacă operația se afișează în consolă sau nu.

Pentru a accesa date publice ale vectorului, se folosesc urmaroarele modalități din fișierul Variabile.cs, din același folder:

* public List<int> Items – proprietate care acceseaza valorile numerice
* public Button this[int i] – indexator pentru controlul care afișează grafic un element al vectorului
* public Size ItemSize – proprietate care setează mărimea tuturor controalelor din vector
* public int AnimationSpeed – proprietate pentru viteza animației
* public int Pause – proprietate pentru durata pauzei dintre animații
* public bool Animating – câmp ce memorează dacă o animație se desfășoară sau nu
* public bool *AutoAnimationMode* – proprietate pentru a seta dacă animațiile se derulează automat sau controlate de la un buton
* public Button NextButton – proprietate care setează butonul care controlează animațiile în caz că proprietatea anterioară e setată la *false,* acest buton fiind exterior controlului de tip VectorControl
* public CheckBox CheckButton – proprietate care setează checkbox-ul care controleză proprietatea *AutoAnimationMode,* fiind exterior controlului
* public TextBox Log – proprietate care setează consola în care se afișează succesiunea operațiilor efectuate, exterioară controlului VectorControl

Din punct de vedere al clientului, o facilitate importantă a acestui control este meniul derulant pe care în conține, ce oferă o gamă largă de funcții:

* Editează elementele
* Schimbă aspect – setări generale ale aspectului vectorului, cum ar fi culorile sau mărimea numerelor sau a căsuțelor
* Viteza operațiilor – schimbă **durata pauzei** dintre două animații succesive
* Mărime – schimbă mărimea căsuței, având valori predefinite pentru mărimi standard: mică, mdeie, mare, extra
* Spațiere – schimbă mărimea spațiului dintre 2 căsuțe/elemente consecutive
* Font – schimbă fontul numerelor. Se recomandă schimbarea dimensiunii acestuia din meniul *Schimbă aspect*
* Scrie în consolă – afișează elementele vectorului în consolă
* Copiază – copiază elementele vectorului în memorie
* Lipește – schimbă elementele vectorului curent în cele memorate
* Reîmprospătează – reface vectorul, în caz că o animație eșuează și la finalul acesteia elementele nu apar în ordine succesivă
* Culoare de Selecție – setează culoarea cu care un element poate fi marcat prin click
* Culoare de Fundal – setează culoarea de fundal a vectorului, nu a elementelor sale
* Culoare Text – schimbă culoarea fontului numerelor, dar și culoarea marginii acestora
* Culoare Elemente – setează culoarea de fundal a elementelor vectorului
* Încarcă din fișier – încarcă elementele vectorului din fișierul ales
* Salvează în fișier – salvează elementele vectorului în fișierul ales

1. Cod sursă

Vom da ca exemplu funcțiile Swap\_One și Swap din Animatii.cs respectiv STL.cs

private void Swap\_One(int i, int j)

{

if (i == j) ;

int aux;

if (i > j) { aux = i; i = j; j = aux; }

Color ci = this[i].BackColor, cj = this[j].BackColor;

this[i].BackColor = OwnColors.Select;

this[j].BackColor = OwnColors.Select;

this[j].BringToFront();

this[i].BringToFront();

for (int k = 0; k < Items.Count; ++k)

if (i != k && j != k)

this[k].SendToBack();

Animating = true;

//animatie cu timer, trebuie ActionQueue ca sa nu se suprapuna...

Log.Text += "swap(" + i + ", " + j + "): " + v[i] + " " + v[j];

Log.AppendText(Environment.NewLine);

System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();

currentTimer = timer;

timer.Tag = string.Format("swap {0} {1}", i, j);

int xi = this[i].Location.X, xj = this[j].Location.X;

timer.Interval = AnimationSpeed \* (ItemSize.Height / 5) /

(Math.Abs(j - i) == 1 ? 2 : Math.Abs(j - i));

int pauze = Math.Max(Pause / timer.Interval, 1);

bool InPauza = false;

timer.Tick += delegate (object sender, EventArgs args) {

if (InPauza)

{

--pauze;

if (pauze <= 0)

{

timer.Stop();

Animating = false;

if (AutoAnimationMode) ProceedNextInQueue();

}

}

else

{

if (this[i].Left < xj) this[i].Left += AnimationSpeed \*

Math.Max(this[i].Width / 50, 1);

if (this[j].Left > xi) this[j].Left -= AnimationSpeed \*

Math.Max(this[j].Width / 50, 1);

if (this[i].Left >= xj && this[j].Left <= xi)

{

this[i].Left = xj; this[j].Left = xi;

Button auxb = this[i]; this[i] = this[j]; this[j] = auxb;

aux = Items[i]; Items[i] = Items[j]; Items[j] = aux;

this[i].BackColor = cj;

this[j].BackColor = ci;

InPauza = true;

ReOrder();

}

}

};

timer.Start();

}

public void Swap(int i, int j)

{

++nrswaps;

if (i < 0 || j < 0 || i == j) return;

if (!Animating && AnimationQueue.Count==0) Swap\_One(i, j);

else AnimationQueue.Enqueue(() => Swap\_One(i, j));

}

Pentru a putea construi vectori cu elemente cu oricât de multe cifre, folosim funcția *getsize(Button b)* membră a clasei *VectorControl* din fișierul *Variabile.cs* pentru a redimensiona lățimea butonului în funcție de numărul de cifre.

Size getsize(Button b)

{

b.Text = b.Text.Trim();

int r = ItemSize.Width;

r += (int)(b.Font.SizeInPoints \* (b.Text.Length - 1));

if (b.Text[0] == '-') r -= (int)(b.Font.SizeInPoints / 3);

return new Size(r, ItemSize.Height);

}

Funcția de redimensionare a tab-ului de prezentare a sortării directe din fișierul *FormMain.Resize.cs* are codul:

void tab5()

{

splitContainer6.Height = tabPage5.Height - flowLayoutPanel5.Top -

flowLayoutPanel5.Height - 10;

flowLayoutPanel5.Width = tabPage5.Width - flowLayoutPanel5.Left - 50;

textBox17.Width = flowLayoutPanel5.Width;

richTextBox8.Height = Math.Max(splitContainer6.Height - 39, button21.Bottom);

richTextBox8.Width = splitContainer6.Panel1.Width - richTextBox8.Left - 10;

textBox10.Height = textBox9.Height = splitContainer6.Height - 39 - textBox10.Top;

if (richTextBox8.Width >= 500) LoadFile(richTextBox8, "sortd2.rtf");

else if (richTextBox8.Width >= 370) LoadFile(richTextBox8, "sortd1.rtf");

else LoadFile(richTextBox8, "sortd.rtf");

label36.Text = "Consolă intershimbări";

if (label36.Width > splitContainer7.Panel1.Width) label36.Text = "Interschimbări";

label34.Text = "Consolă comparări";

if (label34.Width > splitContainer7.Panel2.Width) label34.Text = "Comparări";

if (textBox10.Width + 10 > splitContainer7.Panel1.Width)

textBox10.Width = textBox9.Width = splitContainer7.Panel1.Width - 10;

Center(label36, label34, textBox9, textBox10);

}

Funcția de redimensionare a tab-ului de prezentare a algoritmului de ștergere multiplă din fișierul *FormMain.Resize.cs*, care conține mai multe controale este de forma:

void tab1()

{

// insert/remove simplu

int height\_spacing = 20;

if (this.Width <= 1199)

{

splitContainer1.Font = FntSize(this.Font, 16);

button2.Font = FntSize(button2.Font, 12); button1.Image = null;

button3.Font = FntSize(button3.Font, 12); button4.Image = null;

button1.Font = button4.Font = FntSize(this.Font, 14);

height\_spacing = 10;

vs1.ItemSize = vsi.ItemSize = vi1.ItemSize = vii.ItemSize = new Size(30, 30);

vs1.Font = vsi.Font = vs1.FntSize(13);

vi1.Font = vii.Font = vi1.FntSize(13);

vs1.Left = label2.Right + height\_spacing / 2;

vsi.Left = label3.Right + height\_spacing / 2;

vi1.Left = label11.Right + height\_spacing / 2;

vii.Left = label10.Right + height\_spacing / 2;

}

else

{

splitContainer1.Font = FntSize(this.Font, 20.25f);

button2.Font = FntSize(button2.Font, 16);

button3.Font = FntSize(button3.Font, 16);

button1.Font = button4.Font = FntSize(this.Font, 16);

vs1.ItemSize = vsi.ItemSize = vi1.ItemSize = vii.ItemSize = new Size(40, 40);

vs1.Font = vsi.Font = vs1.FntSize(16);

button1.Image = Properties.Resources.minus;

vi1.Font = vii.Font = vi1.FntSize(16); button4.Image = Properties.Resources.add;

vs1.Left = label2.Right + height\_spacing / 2;

vsi.Left = label3.Right + height\_spacing / 2;

vi1.Left = label11.Right + height\_spacing / 2;

vii.Left = label10.Right + height\_spacing / 2;

}

numericUpDown1.Left = label5.Right + height\_spacing ;

button1.Left = numericUpDown1.Right + height\_spacing ;

numericUpDown2.Left = label8.Right + height\_spacing / 4;

label13.Left = numericUpDown2.Right + height\_spacing / 4;

numericUpDown3.Left = label13.Right + height\_spacing / 4;

button4.Left = numericUpDown3.Right + height\_spacing / 4;

splitContainer11.Height = splitContainer12.Height = splitContainer1.Panel1.Height - (numericUpDown1.Bottom + height\_spacing/2);

if (splitContainer11.Top + 2\*textBox1.Bottom+ height\_spacing/2 < splitContainer1.Panel1.Height)

{

//stanga

splitContainer11.Orientation = Orientation.Horizontal;

splitContainer11.FixedPanel = FixedPanel.None;

splitContainer11.SplitterDistance = splitContainer11.Height/2;

textBox1.Width = splitContainer11.Panel1.Width - 2\*height\_spacing;

richTextBox1.Width = splitContainer11.Panel2.Width - 2\*height\_spacing;

Center(textBox1, richTextBox1);

label4.Left = textBox1.Left;

label6.Left = richTextBox1.Left;

LoadFile(richTextBox1, "stergereM.rtf");

// dreapta

splitContainer12.Orientation = Orientation.Horizontal;

splitContainer12.FixedPanel = FixedPanel.None;

splitContainer12.SplitterDistance = splitContainer12.Height / 2;

textBox2.Width = splitContainer12.Panel1.Width - 2 \* height\_spacing;

richTextBox2.Width = splitContainer12.Panel2.Width - 2 \* height\_spacing;

Center(textBox2, richTextBox2);

label9.Left = textBox2.Left;

label7.Left = richTextBox2.Left;

LoadFile(richTextBox2, "inserareM.rtf");

}

else

{

splitContainer11.SplitterDistance = splitContainer11.Width - splitContainer11.SplitterWidth -( height\_spacing + CopyControl[richTextBox1.Name].Width);

splitContainer12.SplitterDistance = splitContainer12.Width - splitContainer12.SplitterWidth - (height\_spacing + CopyControl[richTextBox2.Name].Width);

//stanga

splitContainer11.Orientation = Orientation.Vertical;

splitContainer11.FixedPanel = FixedPanel.Panel2;

textBox1.Width = splitContainer11.Panel1.Width - height\_spacing;

richTextBox1.Width = splitContainer11.Panel2.Width - height\_spacing;

Center(textBox1, richTextBox1);

label4.Left = textBox1.Left;

label6.Left = richTextBox1.Left;

LoadFile(richTextBox1, "stergere.rtf");

// dreapta

splitContainer12.Orientation = Orientation.Vertical;

splitContainer12.FixedPanel = FixedPanel.Panel2;

textBox2.Width = splitContainer12.Panel1.Width - height\_spacing;

richTextBox2.Width = splitContainer12.Panel2.Width - height\_spacing;

Center(textBox2, richTextBox2);

label9.Left = textBox2.Left;

label7.Left = richTextBox2.Left;

LoadFile(richTextBox2, "inserare.rtf");

}

}

După ce se definesc funcții pentru fiecare tab, apelul lor se face atât în momentul redimensionării formei, cât și la schimbarea tab-ului, pentru a se actualiza design-ul tab-ului. În caz contrar, controalele vor rămâne redimensionate și repoziționate după fostele dimensiuni ale ferestrei, fapt ce dă impresia a bug:

private void FormMain\_Resize(object sender, EventArgs e) => tabControl1\_SelectedIndexChanged(sender, e);

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (tabControl1.SelectedIndex == 0) tab0(); // editor

else if (tabControl1.SelectedIndex == 1) tab1(); // insert/remove simplu

else if (tabControl1.SelectedIndex == 2) tab2(); // stergere multipla

else if (tabControl1.SelectedIndex == 3) tab3(); // inserare multipla

else if (tabControl1.SelectedIndex == 4) tab4(); // sort indirecta

else if (tabControl1.SelectedIndex == 5) tab5(); // sort directa

else if (tabControl1.SelectedIndex == 6) tab6(); // sort bule

else if (tabControl1.SelectedIndex == 7) tab7(); // insert sort

}

1. Manual de utilizare

Facilități generale

O facilitate importantă a acestei aplicații este redimensionarea. Aceasta este optimizată pentru ecrane cu rezoluțiile peste 1025 x 600 și este testată pănă la 1920 x 1024 pixeli. La ecranele sub 1025 x 600, fereastra va fi mai largă decât ecranul. Cât despre ecranele mai mari, este posibil ca textul să aibă un font puțin mai mic, fiind mai greu de citit.

O altă facilitate importantă este existența indicațiilor în interiorul aplicației. Astfel, pentru a înțelege mai bine rolul unui buton, al unui meniu, al unui text, etc. ținera curosrului asupra obiectului respectiv poate determina apariția unei indicații.

Arhitectura unui tab

Aplicația Vectori este alcătuită din mai multe tab-uri, care se pot schimba dând click pe titlurile din bara din partea de sus a aplicației.

Un tab conține, în general, următoarele componente:

* Descrierea în limbaj natural al algoritmului prezentat
* Codul sursă c++, care stă la baza animației
* Vectorul reprezentat grafic, editabil ca aspect și valori
* Consolă de operații, în care se înregistrează operațiile efectuate

De menționat că codul sursă nu poate fi modificat, iar consola de operații poate fi modificată, dar nu se pot utiliza caracterele, numai cele de control, precum tasta BACKSPACE, SPACE, săgețile, etc.

Rularea unei animații

Butonul Execută determină începerea rulării animației codului din imediata sa apropiere.

Butonul Reset șterge textul din consolele, anulează animația și restabilește valorile inițiale ale vectorului suport. Dacă o animație nu s-a terminat de efectuat, resetarea se va executa când animația se va opri.

Căsuța Auto Play marchează dacă animația rulează singură sau este controlată de butonul Next Step. Modul poate fi schimbat în timpul rulării. Efectele butonului Next Step nu se stachează, și nu au efect când căsuța Auto Play este bifată.

Semnificația variabilelor din cod

v - vectorul

n - numărul de elemente al vectorului

i, j - indexi ai vectorului

p - poziția la care se face operația

Semnificația comenzilor din consolă

erase(i): v – șterge valoarea v de la poziția i

insert(i): v – inserează valoarea v la poziția i

add: v – adaugă valoarea v, la finalul vectorului

swap(i, j): x y – interschimbă valorile x și y aflate la indicii i, respectiv j

Culorile

* Culoarea de Selecție
* Culoarea de Fundal
* Culoarea Textului
* Culoarea Elementelor
* Culoarea de Inserție
* Culoarea de Ștergere
* Culoarea de Marcare

Prin apăsarea unui element al vectorului, acesta își va schimba culoarea de fundal într-una specifică, numită **culoare de selecție**. La apăsarea elementului pentru a doua oară, aseastă culoare va dispărea. Culoarea prestabilită e galben.

Culoarea de fundal reprezintă culoarea de pe fundalul unui vector, mai precis a spațiului dintre elemente, nu al elementelor sale.

Culoarea textului reprezintă culoarea cu care vor fi scrise numerele din interiorul unui vector, dar și a conturului elementelor. Prestabilită e culoarea portocaliu.

Culoarea elementelor reprezintă culoarea de fundal al elementelor din vector, posibil confundabilă cu culoarea de fundal. La schimbarea acestei proprietăți, elementele cu fundalul modificat prin marcare nu se vor schimba decât prin apăsarea butonului reîmprospătează.

Aceste culori pot fi schimbate din meniul derulant al aplicației, al căror icoane reprezintă culoarea setată lor.

Culoarea de inserție reprezintă culoarea cu care se marchează elementele noi care se inserează în vector. Prestabilită e culoarea verde.

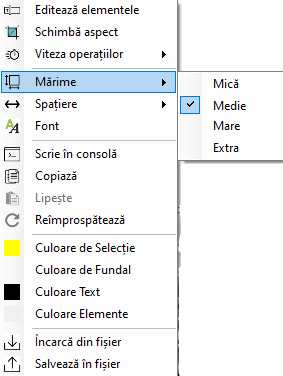
Culoarea de ștergere reprezintă culoarea cu care se marchează elementele care se șterg din vector. Prestabilită e culoarea roșu.

Culoarea de marcare reprezintă culoarea cu care se marchează o secvență de elemente în vector. Prestabilită e culoarea portocaliu.

Aceste ultime 3 culori se pot modifica din tab-ul Editor, subtab-ul Culori al meniului de opțiuni.

Recomandare: pentru a înțelege mai bine aceste culori, este utilă deschiderea și testarea lor în meniul Schimbă aspect din meniul derulant al unui vector.

De asemenea, vectorul beneficiază de un meniu derulant amplu.



Editor

Acest tab este primul deoarece acesta concentrează toată aplicația. Este alcătuit din 3 regiuni:

* vectorul propriu-zis, aflat în partea dreaptă
* meniul de operații, aflat în partea stângă
* consola operațiilor, aflată în partea de jos

Un aspect important este că aceste reginui sunt redimensionabile prin apăsarea spațiului liber dintre acestea.

Celelalte tab-uri, în afară de Ajutor, respectă arhitectra prezentată mai sus, și nu le vom prezenta în particular, iar tab-ul Ajutor prezintă întocmai manualul de utilizare curent.

1. Bibliografie

Link-uri folosite pentru descrierea părții teoretice:

* <https://www.pbinfo.ro/articole/5584/stergeri-si-inserari-de-elemente>
* <https://www.pbinfo.ro/articole/5589/metoda-bulelor>
* <https://www.pbinfo.ro/articole/5605/sortarea-prin-selectie>
* <https://www.pbinfo.ro/articole/5609/sortarea-prin-insertie>

Link-uri folosite în documentarea pentru creerea aplicației:

* <https://stackoverflow.com/questions/6102241/how-can-i-add-moving-effects-to-my-controls-in-c>
* <https://docs.microsoft.com/en-us/aspnet/web-forms/overview/ajax-control-toolkit/animation/adding-animation-to-a-control-cs>