DATU STRUKTŪRAS UN ALGORITMI
Praktisko darbu uzdevumi

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.1

Darba nosaukums: Rekursīvās funkcijas.

Laiks: 2 mācību stundas.

Darba mērķis: Iemācīties veidot rekursīvus algoritmus.

Darba gaita: Programmēšanas vidē C++ Builder izpildīt šādus uzdevumus:

1. Aprakstiet rekursīvu un iteratīvu algoritmus, kuri aprēķinātu:
    a. 1. variants - **N.** pēc kārtas Fibonači skaitli;
    b. 2. variants - naturālā skaitļa faktoriālu **N!**;
2. Aprakstiet rekursīvu funkciju, kura sareizina divus naturālus skaitļus neizmantojot reizināšanas operāciju;
3. Aprakstiet iteratīvu funkciju, kura pārbaudītu, vai dotais skaitlis ir pirmskaitlis;

Fibonači funkcija:
https://en.wikibooks.org/wiki/Algorithm_Implementation/Mathematics/Fibonacci_Number_Program
Faktoriāla funkcija:
http://rosettacode.org/wiki/Factorial

Vērtēšanas tabula

Uzdēvums	1. Rekursīvs algoritms	1. Iteratīvs algoritms	2. uzdevums	3.uzdevums
Punktu skaits	*līdz 2*	*līdz 2*	*līdz 3*	*līdz 3*

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.2

Darba nosaukums: Kārtošanas algoritmi.

Laiks: 2 mācību stundas.

Darba mērķis: Iemācīties izmantot kārtošanas algoritmus praktiskajos uzdevumos.

Darba gaita: Programmēšanas vidē C++ Builder izpildīt šādus uzdevumus:

1. Salīdzināt algoritmu Bubble Sort, Insertion Sort un Selection Sort ātrdarbību, ja elementu skaits ir 5 000, 10 000, 20 000, 40 000 un 80 000. Kārtošanas laiku katram algoritmam attēlot atsevišķajos ListBox komponentos.

2. Ar Bubble Sort algoritma palīdzību realizēt simbolu sakārtošanu vārdā, kas ir ierakstīts Edit laukumā;

3. Ar Selection Sort algoritma palīdzību realizēt teksta rindu masīvu sakārtošanu;

Piezīme: Laika nomērīšanai izmantojiet funkciju:
1.variants - GetTickCount();
2.variants - Now().

Vērtēšanas tabula

Uzdēvums	1. uzdevums	2. uzdevums	3.uzdevums
Punktu skaits	*līdz 4*	*līdz 3*	*līdz 3*

// SELECTION SORT algoritms

```
void selectionSort(int arr[], int n) {
      int i, j, minIndex, tmp;
      for (i = 0; i < n - 1; i++) {
            minIndex = i;
            for (j = i + 1; j < n; j++)
                  if (arr[j] < arr[minIndex])
                        minIndex = j;
            if (minIndex != i) {
                  tmp = arr[i];
                  arr[i] = arr[minIndex];
                  arr[minIndex] = tmp;
            }
      }
}
```

// INSERTION SORT algoritms

```
void insertionSort(int arr[], int length) {
    int i, j, tmp;
    for (i = 1; i < length; i++) {
        j = i;
        while (j > 0 && arr[j - 1] > arr[j]) {
            tmp = arr[j];
            arr[j] = arr[j - 1];
            arr[j - 1] = tmp;
            j--;
        }
    }
}
```

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.3

Darba nosaukums: Meklēšanas algoritmi.

Laiks: 2 mācību stundas.

Darba mērķis: Iemācīties izmantot meklēšanas algoritmus praktiskajos uzdevumos.

Darba gaita: Programmēšanas vidē C++ Builder izpildīt šādus uzdevumus:

4. Izveidot veselo skaitļu masīvu no 10 000 elementiem, aizpildīt to ar gadījuma skaitļiem (diapazonā no –50 000 līdz 50 000). Izmantojot lineāru meklēšanu, aprēķināt meklēšanas laiku pēc N izmēģinājumiem (100<=N<=1000). Izmēģinājumu skaits un meklējamā vērtība jāievadā programmas izpildes laikā. Izvadīt pozitīvo meklēšanas rezultātu skaitu.

5. Sakārtot masīvu, izmantojot izvēles, iestarpināšanas vai citu kārtošanas algoritmu. Izmantojot bināro meklēšanu, aprēķināt meklēšanas laiku pēc N izmēģinājumiem (100<=N<=1000). Izmēģinājumu skaits un meklējamā vērtība jāievadā programmas izpildes laikā. Izvadīt pozitīvo meklēšanas rezultātu skaitu.

6. Pilnveidot 2. uzdevumu: aprēķināt meklēšanas algoritma vidējo salīdzinājumu skaitu.

Vērtēšanas tabula

Uzdēvums	1. uzdevums	2. uzdevums	3.uzdevums
Punktu skaits	*līdz 4*	*līdz 3*	*līdz 3*

Bināras meklēšanas funkcija ar izmantošanas piemēru:

```
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        // If the element is present at the middle
        // itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, r, x);
    }

    // We reach here when element is not
    // present in array
    return -1;
}

int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? cout << "Element is not present in array"
                   : cout << "Element is present at index " << result;
    return 0;
}
```

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.4

Darba nosaukums: Sakārtoto sarakstu izveide

Laiks: 2 mācību stundas.

Darba mērķis: Iemācīties izmantot saistītus sarakstus praktiskajos uzdevumos.

Darba gaita: Programmēšanas valodā C++ realizēt šādus uzdevumus:

1. Izveidot veselo skaitļu sarakstu:
    a. Organizēt 5 saraksta mezglu vērtību ievadi no tastatūras;
    b. Papildināt sarakstu ar 3 jauniem mezgliem, izmantojot gadījuma vērtības;
    c. Pieprasīt ievadīt mezgla kārtas numuru, izvadīt atbilstošo elementu uz ekrāna. Izdzēst to.
    d. Izvadīt visas saraksta vērtības uz ekrāna. Izdzēst sarakstu.
2. Izveidot dubultsaišu simbolu sarakstu:
    a. Izveidot 5 saraksta mezglus ar patvaļīgām vērtībām;
    b. Izvadīt visu mezglu vērtības tiešā un pretējā secībā;
    c. Papildināt sarakstu ar diviem jauniem mezgliem, kuru vērtības ir Jūsu vārda un uzvārda pirmie burti. Ievietošanas pozīcijas jāievadā no tastatūras;
    d. Izvadīt visas saraksta vērtības uz ekrāna. Izdzēst sarakstu.
3. Pilnveidot 1.uzdevumu tā, lai katrā mezglā būtu iespēja glabāt 3 veselos skaitļus.

Vērtēšanas tabula

Uzdēvums	1.a	1.b	1.c	1.d	2.a	2.b	2.c	2.d	3.
Punktu skaits	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*2*

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.5

Darba nosaukums: Darbs ar stekiem un rindām

Laiks: 4 mācību stundas.

Darba mērķis: Iemācīties izmantot stekus un rindas praktiskajos uzdevumos.

Darba gaita: Programmēšanas valodā C++ realizēt šādus uzdevumus:

1.-2. stundas:

1. Nodefinēt 3 veselo skaitļu stekus: **digits**, **positive** un **negative**.
   Pieprasīt ievadīt no tastatūras 10 veselos skaitļus. Saglabāt tos stekā **digits**. Visus pozitīvus skaitļus no steka **digits** pārvietot stekā **positive**, bet negatīvus stekā **negative**.
2. Izvadīt uz ekrāna visus steka **positive** skaitļus tiešā un apgrieztā secībā.
3. Nodefinēt simbolu steku **symbols**. Ar cikla palīdzību saglabāt stekā ievadītos no tastatūras simbolus kamēr netiks ievadīts simbols *****. Noteikt un izvadīt uz ekrāna steka **symbols** vidus elementu.
4. Saglabāt stekā **symbols** 100 gadījuma simbolus. Izdzēst no steka visus lielus latiņu burtus.

Piezīme: uzdevumu izpildei ir aizliegts izmantot masīvus!!!

3.-4. stundas:

5. Nodefinēt simbolu rindu **symbols**. Ar cikla palīdzību saglabāt rindā ievadītos no tastatūras simbolus kamēr netiks ievadīts simbols *****.
6. Izvadīt uz ekrāna rindas **symbols** elementus tiešā un pretējā secībā
7. Izņemt no rindas **symbols** visus ciparus. Izvadīt rindas elementus uz ekrāna.
8. Saskaitīt, cik ir lielo latiņu burtu rindā **symbols**. Pēc burtu skaitīšanas rindas saturam jābūt tādam pašam, kā pirms skaitīšanas.
9. Mainīt vietām rindas **symbols** pirmo un pēdējo elementu. Izvadīt rindas elementus uz ekrāna.

Vērtēšanas tabula

Uzdēvums	1.	2.	3.	4.	5.	6.	7.	8.	9
Punktu skaits	*2*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*

Datu struktūras un algoritmi.
VII semestris.
Praktiskais darbs Nr.6

Darba nosaukums: Šifrēšanas algoritmi

Laiks: 2 mācību stundas.

Darba mērķis: Iemācīties izmantot vienkāršus šifrēšanas algoritmus praktiskajos uzdevumos.

Darba gaita: Programmēšanas vidē Delphi realizēt sekojošus uzdevumus:

1. Cēzara algoritms:
    a. Realizēt simbolu virknes iešifrēšanas funkciju pēc Cēzara algoritma. Funkcijai jāpieņem 2 parametri – simbolu virkne un nobīde.
    b. Realizēt simbolu virknes dešifrēšanas funkciju.
    c. Nodrošināt iespēju izmantot jebkāda izmēra nobīdi.
2. Viženera algoritms:
    a. Realizēt simbolu virknes iešifrēšanas funkciju pēc Viženera algoritma. Funkcijai jāpieņem 2 parametri – simbolu virkne un maska.
    b. Realizēt simbolu virknes dešifrēšanas funkciju.

Paredzēt programmā iespēju pārbaudīt iešifrēšanas/dešifrēšanas funkcijas darbību.

Vērtēšanas tabula

Uzdēvums	1.a	1.b	1.c	2.a	2.b
Punktu skaits	*2*	*2*	*2*	*2*	*2*