Міністерство освіти і науки України

Волинський національний університет імені Лесі Українки

Кафедра комп’ютерних наук та кібербезпеки

Т.О. Гришанович

ЛАБОРАТОРНИЙ ПРАКТИКУМ ІЗ ДИСЦИПЛІНИ “АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ”

для студентів спеціальності 122 Комп’ютерні науки
першого (бакалаврського) рівня

Луцьк 2021

УДК 004.421:004.422.63(076)
Г 85

Рекомендовано до видання науково-методичною радою
Волинського національного університету імені Лесі Українки
(протокол № 4 від 14 грудня 2021 р.)

Рецензенти:
Собчук О. М. – кандидат педагогічних наук, доцент, доцент кафедри загальної математики та методики навчання інформатики Волинського національного університету імені Лесі Українки;
Ліщина Н. М. – кандидат технічних наук, доцент, завідувач кафедри комп'ютерних наук Луцького національного технічного університету.

Лабораторний практикум з дисципліни «Алгоритми та структури даних» для студентів спеціальності 122 Комп'ютерні науки [Електронний ресурс] / Т.О. Гришанович; ВНУ імені Лесі Українки. Електронні текстові данні (1 файл: 859 КБ). Луцьк : ВНУ імені Лесі Українки, 2021. – 50 с.

Лабораторний практикум містить інструкції до виконання лабораторних робіт із дисципліни «Алгоритми та структури даних». Для кожної роботи наведено тему та мету, на прикладі продемонстровано хід її виконання, наведено зразки вхідних/вихідних даних. Читачу запропоновано перелік джерел до кожної із робіт, аби він міг не тільки здійснити підготовку до виконання, але й поглиблено розглянути тему. Практикум також містить ряд завдань для індивідуального виконання, які не мають прив'язки до мови програмування і можуть бути запрограмовані із використанням засобів довільних мов.
Лабораторний практикум призначений для здобувачів першого (бакалаврського) ступеня, що навчаються за спеціальністю 122 Комп'ютерні науки.

Зміст

Лабораторна робота 1. Найпростіші алгоритми. Цілочисельна арифметика

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що оперують цілими числами та використовують для їх обробки арифметичні операції, такі як множення, додавання, віднімання та операцію присвоювання.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми для піднесення чисел до заданого степеня без використання операції піднесення до степеня;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Приклад: Дано ціле число x. Користуючись тільки операцією множення, отримати:

Завдання. Дано ціле число x. Користуючись тільки операцією множення, отримати x^9 за чотири операції.

Алгоритм виконання:

1. Початок.
2. Вхід: x
3. a = x*x // отримуємо x у степені 2
4. b = a*a // отримуємо x у степені 4
5. c = b*b // отримуємо x у степені 8
6. d = c*x // отримуємо x у степені 9
7. Вихід: d, що за значенням рівно x^9.

Реалізація алгоритму на мові програмування:

```
# include <iostream>
using namespace std;
int main(){
   int x;
   cout<<"Input x";
   cin>>x;
   int a = x*x;
   int b = a*a;
   int c = b*b;
```

```
    int d = c*x;
    cout<<d;
return 0;
}
```

Результат роботи програми:

1) Input x 2

   512

2) Input x 3

   19 683

3) Input x 4

   262 144

Завдання для індивідуального виконання:

Знайти значення величини

1. x^8 за три операції
2. x^{17} за п’ять операції
3. x^{10} за чотири операції
4. x^{13} за п’ять операцій
5. x^{21} за шість операції
6. x^{15} за п’ять операцій
7. x^{28} за шість операції
8. x^{64} за шість операції
9. x^{10} за чотири операції
10. x^{20} за п’ять операцій

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Ільман В. М., Іванов О. П., Панік Л. О. Алгоритми, дані і структури : навч. посіб. / Дніпро : Дніпропет. нац. ун-т залізн. трансп.ім. акад. В. Лазаряна, 2019. 134 с.
2. Шаховська Н. Б., Голощук Р. О. Алгоритми і структури даних. Навчальний посібник. Львів : Магнолія, 2018. 216 с

Лабораторна робота 2. Цілочисельна арифметика

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що оперують цілими числами та використовують для їх обробки арифметичні операції, такі як множення, додавання, віднімання, ділення націло, отримання остачі від ділення та операцію присвоювання.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми для виділення цифр у числі, зокрема, отримувати кількість одиниць, десятків, сотень і т.д. у числах, для яких наперед вказано кількість розрядів;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Дано тризначне число *n*. Отримати число, яке утворюється при перестановці 1 та 3 цифр. Наприклад: 234 — 432.

Алгоритм виконання:

1. Початок
2. Вхід: число n
3. a = n%10 // отримуємо число одиниць числа n
4. b = n/100 // отримуємо число сотень числа n
5. c = (n%100)/10 // отримуємо число десятків числа n
6. N = a*100 + c*10 + b // утворюємо число, у якому кількість сотень рівна кількості одиниць числа n, кількість десятків — кількості десятків числа n, кількість одиниць — кількості сотень числа n
7. Вихід: N
8. Кінець

Реалізація на мові програмування:

```
# include <iostream>
using namespace std;
int main(){
   int n;
   cout<<"Input n";
   cin>>n;
   int a = n%10;
```

```
    int b = n/100;
    int c = (n%100)/10;
    int N = a*100 + c*10 + b;
    cout<<N;
return 0;
}
```

Результат роботи програми:

1) Input x 512

   215

2) Input x 703

   307

3) Input x 948

   849

Завдання для індивідуального виконання:

1. Дано тризначне число *n*. Отримати число, яке утворюється при прочитанні числа *n* з права на ліво. 234 — 432.
2. Дано тризначне число *n*. У ньому забрали першу цифру та дописали її вкінці. Яке число отримали в результаті? Наприклад: 234 — 342.
3. Дано тризначне число *n*. Знайти число десятків та одиниць у ньому.
4. Дано тризначне число *n*. Отримати число, яке утворюється при перестановці 2 та 3 цифр. Наприклад: 234 — 243.
5. Дано чотиризначне число *n*. Отримати число, яке утворюється при перестановці 2 та 3 цифр. Наприклад: 2345 — 2435.
6. Дано тризначне число *n*. Отримати число, яке утворюється при перестановці 1 та 2 цифр. Наприклад: 234 — 324.
7. Дано чотиризначне число *n*. Отримати число, яке утворюється при прочитанні числа *n* з права на ліво. 2345 — 5432.
8. Дано тризначне число *n*, в якому усі цифри різні. Отримати 6 чисел, які утворюються при перестановці цифр даного числа.
9. Дано чотиризначне число *n*. Знайти число сотень та десятків у ньому.
10. Дано чотиризначне число *n*. Отримати число, яке утворюється при перестановці 1 та 4 цифр. Наприклад: 2345 — 5342.

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123.
2. Ільман В. М., Іванов О. П., Панік Л. О. Алгоритми, дані і структури : навч. посіб. / Дніпро : Дніпропет. нац. ун-т залізн. трансп.ім. акад. В. Лазаряна, 2019. 134 с.
3. Шаховська Н. Б., Голощук Р. О. Алгоритми і структури даних. Навчальний посібник. Львів : Магнолія, 2018. 216 с
4. Онищенко В. В., Коник Р. С Алгоритми та структури даних. К : 2017. 66 с.

Лабораторна робота 3. Цілочисельна арифметика. Прикладні задачі

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що оперують цілими числами та використовують для їх обробки арифметичні операції, такі як множення, додавання, віднімання, ділення націло, отримання остачі від ділення та операцію присвоювання.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми для розв'язування прикладних задач із використанням цілочисельної арифметики — множини операцій, що застосовні до даних цілого типу;
- вміти переходити від прикладного формулювання задачі до її математичної моделі;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Дано натуральне число $M \leq 1200$, що минули від деякої події до цього часу. Вивести повну кількість років та місяців, які минули від деякої події до цього часу.

Алгоритм виконання:

1. Початок
2. Вхід: число M
3. y = M/12 // отримуємо кількість повних років, що минула від деякої події
4. m = M%12 // отримуємо кількість місяців, що минула від деякої події
5. Вихід: y, m
6. Кінець

Реалізація на мові програмування:

```
# include <iostream>
using namespace std;
int main(){
   int M;
   cout<<"Input M";
   cin>>M;
```

```
    int y = M/12;
    int m = M/100;
    cout<<y<<" years and "<<m<<" months";
return 0;
}
```

Результат роботи програми:

1) Input M 28

   2 years and 4 months

2) Input M 104

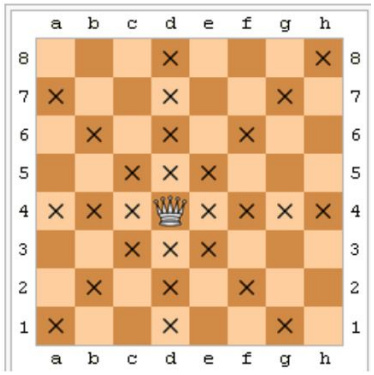
   8 years and 8 months

3) Input M 513

   42 years and 9 months

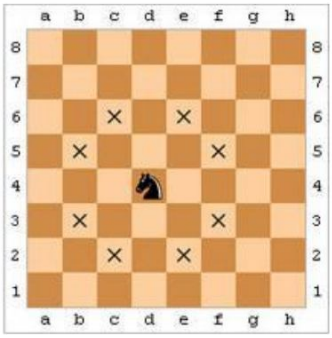
Завдання для індивідуального виконання:

1. Дані цілі числа h, m, s (0<h≤23, 0≤m≤59, 0≤s≤59), які вказують на момент часу h годин, m хвилин, s секунд. Визначити кут (в градусах) між положенням стрілки, яка показує години, на початку доби та у заданий момент часу.
2. Поле шахової дошки задається парою натуральних чисел: перше число – номер вертикалі (рахунок зліва на право), друге число – номер горизонталі (рахунок знизу догори). Дано натуральні числа *k*, *l*, *m*, *n* – усі менші від 8. З'ясувати, чи полю (*k*, *l*) загрожує ферзь, розміщений на полі (*m*, *n*). Можливі ходи ферзя показані на малюнку.



3. Від початку доби минуло n секунд. Визначити, скільки повних годин минуло від початку доби.
4. Дано натуральне число *m* ≤1188, яке є віком людини в місяцях. Вивести повну кількість років та місяців.

5. Поле шахової дошки задається парою натуральних чисел: перше число – номер вертикалі (рахунок зліва на право), друге число – номер горизонталі (рахунок знизу догори). Дано натуральні числа *k*, *l*, *m*, *n* – усі менші від 8. З'ясувати, чи полю (*k*, *l*) загрожує кінь, розміщений на полі (*m*, *n*). Можливі ходи коня показані на малюнку.



6. Дані цілі числа h та m (0<h≤12, 0<m≤60), які показують момент часу h годин m хвилин. Визначити найменший час (кількість повних хвилин), які повинні пройти до того часу, поки хвилинна стрілка та стрілка, яка показує години розмістяться перпендикулярно одна до одної.
7. Від початку доби минуло n секунд. Визначити, скільки повних хвилин минуло від початку останньої години.
8. Від початку доби минуло n секунд. Визначити, скільки секунд минуло від початку останньої хвилини.
9. Поле шахової дошки задається парою натуральних чисел: перше число – номер вертикалі (рахунок зліва на право), друге число – номер горизонталі (рахунок знизу догори). Дано натуральні числа *k*, *l*, *m*, *n* – усі менші від 8. З'ясувати, чи поля (*k*, *l*) та (*m*, *n*) однакового кольору.
10. Задано три натуральних числа A, B і N. Скласти алгоритм знаходження натуральних чисел, які не перевищують N, які можна представити у вигляді суми (довільного числа) доданків, кожне з яких буде A або B.

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123.
2. Богач І. В., Довгалець С. М., Дубовой В, М. Алгоритми розв’язання задач з програмування. Вінниця : ВНТУ, 2017. 119 с.
3. Власій О. О. Алгоритми та структури даних: Лабораторний практикум. Івано-Франківськ : ДВНЗ «Прикарпатський національний університет імені Василя Стефаника», 2015. 68 с.
4. Махровська Н.А., Погромська Г. С. Алгоритми і структури даних: навчально-методичний посібник. Миколаїв : МНУ ім. В.О. Сухомлинського, 2019. 279 с.

Лабораторна робота 4. Розробка алгоритмів із використанням структури розгалуження

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що використовують структуру розгалуження, повного та неповного, вкладених розгалужень, а також формулювання логічних умов, які будуть використані у розгалуженнях.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми із використанням алгоритмічної структури розгалуження;
- вміти формулювати логічні умови для структури розгалуження із використанням логічних операцій;
- вміти переходити від прикладного формулювання задачі до її математичної моделі;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Розв’язати квадратне рівняння виду $ax^2 + bx + c = 0$.

Алгоритм виконання:

1. Початок
2. Вхід: a, b, c.
3. якщо $a = 0$, то
4. якщо $b = 0$, то
5. якщо $c = 0$, то рівняння має безліч розв’язків. Перейти до 13
6. інакше рівняння розв’язку не має. Перейти до 13
7. інакше $x_1 = x_2 = -c/b$. Перейти до 13
8. інакше $D = b^2 - 4ac$
9. якщо $D = 0$, то $x_1 = x_2 = -b/2a$
10. інакше якщо D < 0, то рівняння розв’язку не має
11. інакше $x_1 = -b + sqrt(D)/2a$, $x_2 = -b — sqrt(D)/2a$
12. Вихід: x_1, x_2

13. Кінець

Реалізація на мові програмування:

```
# include <iostream>
using namespace std;
int main(){
   float a, b, c, D, x1, x2;
   cout<<"Input coeficients of the equation"<<endl;
   cout<<"Input a";
   cin>>a;
   cout<<"Input b";
   cin>>b;
   cout<<"Input c";
   cin>>c;
   if (a==0){
     if (b==0){
       if (c==0){
          cout<<"The equation has infinitly many solutions";
       }
       else{
          cout<<"The equation has no solution";
       }
     }
     else{
        x1=-c/b;
        cout<<"x = "<<x1;
     }
   }
   else{
      D=b*b-4*a*c;
      if (D==0){
          x1=-b/(2*a);
          cout<<"x1 = x2 = "<<x1;
      }
      else{
         if (D<0){
           cout<<"The equation has no solution";
          }
         else{
           x1=-b-sqrt(D)/(2*a);
           x2=-b+sqrt(D)/(2*a);
           cout<<"x1 = "<<x1;
           cout<<"x2 = "<<x2;
         }
      }
  }
return 0;
}
```

Результат роботи програми:

1) Input a 0

Input b 0

Input c 0

The equation has infinitly many solutions

2) Input a 0

Input b 0

Input c 24

The equation has no solution

3) Input a 0

Input b 4

Input c 26

x = 4.5

4) Input a 1

Input b -4

Input c 4

x1 = x2 = 2

5) Input a 1

Input b -2

Input c -15

x1 = -3 x2 = 5

6) Input a 3

Input b 2

Input c 15

The equation has no solution

Завдання для індивідуального виконання:

1. Дано тризначне число. Перевірити, чи входить до нього цифра *x*.
2. Рік є високосним, якщо його номер кратний 4. Але із років, кратних 100, високосними є ті, які кратні 400. Дано натуральне число *x*, перевірити, чи *x* — вискосний рік.
3. Перевірити, чи трикутник, який задається точками (*x*1, *y*1), (*x2*, *y2*), (*x3*, *y3*), містить початок координат.

4. Дано чотиризначне число. Перевірити, чи всі його цифри різні.

5. Дано чотиризначне число. Чи правда, що воно містить 3 однакові цифри?
6. Дано чотиризначне число. Чи правда, що воно паліндром? (однаково читається зліва на право, наприклад, 3773 — так, 3777 — ні).
7. Дата задається номером місяця та номером дня. Отримати наступний та попередній дні (номер місяця та номер дня). Рік не високосний.
8. Визначити, чи дане 6-значне число — щасливе (сума перших трьох та сума останніх трьох — однакова).
9. Визначити, чи дане 4-значне число — щасливе (сума перших двох та сума останніх двох — однакова).
10. Дано чотиризначне число. Чи правда, що воно містить 2 однакові цифри?

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123.
2. Богач І. В., Довгалець С. М., Дубовой В, М. Алгоритми розв’язання задач з програмування. Вінниця : ВНТУ, 2017. 119 с.
3. Власій О. О. Алгоритми та структури даних: Лабораторний практикум. Івано-Франківськ : ДВНЗ «Прикарпатський національний університет імені Василя Стефаника», 2015. 68 с.
4. Льовкін В. М. Методичні вказівки до виконання самостійної роботи студентів та розрахунково-графічних завдань з дисципліни “Алгоритмізація та програмування” для студентів спеціальності 122 “Комп’ютерні науки” (всіх форм навчання). Запоріжжя : ЗНТУ, 2017. 54 с.

Лабораторна робота 5. Розробка алгоритмів із використанням структури циклу з параметром

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що використовують структуру циклу з параметром.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми із використанням циклу з параметром;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Обчислити значення скінченного добутку $p = \prod_{i=2}^{n} \frac{i^3 - 2}{(i-3)(i-6)}$.

Алгоритм виконання:

1. Початок
2. Вхід: *n*
3. i, p = 1
4. для i від 2 до n виконати
5. якщо i = 3 або i = 6 перейти до 7
6. інакше p = p* (i^3 - 2)/((i-3)*(i-6))
7. i = i+1
8. Вихід: p
9. Кінець

Реалізація на мові програмування:

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
  int i,n;
  float p=1.0;
  cout<<"Input n: "<<endl;
  cin>>n;
  for (i=2; i<=n; i++) {
     if ((i- 3)==0 || (i- 6)==0) {
       continue;
     }
     p= p* (pow(i,3)- 2)/((i- 3)*(i- 6));
```

```
  }
  cout<<" "<<p<<endl;
return 0;
}
```

Результат роботи програми:

1) Input n 4

-46.5

2) Input n 7

243794

Завдання для індивідуального виконання: Розробити алгоритм для обчислення скінченної суми.

1. $s=\sum_{i=0}^{n}\frac{5i^2+2i-3}{(i-1)(i-4)}$

2. $s=\sum_{i=1}^{n}\frac{4i^2+3i-1}{(i-3)(i-5)}$

3. $s=\sum_{i=2}^{n}\frac{5i^3-2i+4}{(i-2)(i-6)}$

4. $s=\sum_{i=1}^{n}\frac{5i^2+2i+3}{(i-1)(i-4)}$

5. $s=\sum_{i=0}^{n}\frac{i^2+2i+7}{(i-3)(i-6)}$

6. $s=\sum_{i=-4}^{n}\frac{i^2+5i+3}{(i-3)(i+4)}$

7. $s=\sum_{i=-2}^{n}\frac{3i^2+7i+4}{(i+1)(i-2)}$

8. $s=\sum_{i=-3}^{n}\frac{i^2+4i+1}{(i-1)(i+2)}$

9. $s=\sum_{i=-2}^{n}\frac{i^2+8i+3}{(i+1)(i-4)}$

10. $s=\sum_{i=-4}^{n}\frac{i^2+3i+9}{(i+5)(i-7)}$

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Богач І. В., Довгалець С. М., Дубовой В, М. Алгоритми розв’язання задач з програмування. Вінниця : ВНТУ, 2017. 119 с.
2. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123.
3. Дудзяний І. М. Програмування мовою С++. Частина 1 : Парадигма процедурного програмування : навчальний посібник. Львів : ЛНУ імені Івана Франка, 2013. 468 с.

Лабораторна робота 6. Розробка алгоритмів із використанням вкладених циклів із параметром

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що використовують вкладені цикли з параметром.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми із використанням вкладених циклів, зокрема циклів з параметром;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Дано натуральні числа m та n. Розробити алгоритм та програму для обчислення суми $1^n + 2^n + 3^n + ... + m^n$. Операцію піднесення до степеня не використовувати.

Алгоритм виконання:

1. Початок
2. Вхід: *m, n*
3. *S* = 0 // змінна для зберігання результату
4. x // проміжна змінна
5. i, j // змінні, які будуть використані у циклах
6. для i від 1 до m виконати
7. x = 1
8. для j від 1 до n виконати
9. x = x* i // множимо значення і само на себе n разів
10. j = j + 1
11. S = S +x
12. i = i+1
13. Вихід: S
14. Кінець

Реалізація на мові програмування:

```
#include <iostream>
using namespace std;
int main(){
   int m,n,x, S = 0;
```

```
    cout<<"Input m";
    cin>>m;
    cout<<"Input n";
    cin>>n;
    for (int i=1; i<=m; i++){
       x=1;
       for (int j=1; j<=n; j++){
          x *=i;
       }
    S +=x;
    }
    cout<<S;
return 0;
}
```

Результат роботи програми:

1) Input m 2

Input n 4

30

1) Input m 3

Input n 5

276

Завдання для індивідуального виконання:

1. Дано натуральне число n (n≤ 27). Знайти всі тризначні числа, сума цифр яких рівна n. Операції ділення, цілочисельного ділення та отримання остачі від ділення не використовувати.
2. Знайти всі цілі числа з проміжку від 1 до 300, у яких рівно п'ять дільників.
3. Знайти всі цілі числа з проміжку від 200 до 500, у яких рівно шість дільників.
4. Знайти всі цілі числа з проміжку від a до b, у яких рівно k дільників.
5. Дано натуральні числа m та n. Знайти $1^n + 2^n + 3^n + \ + m^n$.
6. Знайти усі прості нескоротні дроби із проміжку [0; 1], знаменник яких не перевищує 7. (дріб задається двома числами – чисельником та знаменником. Наприклад, 2/7 – простий нескоротний дріб).
7. Дано натуральне число a. Отримати усі його дільники.
8. Знайти всі цілі числа з проміжку від 1 до 300, у яких рівно п’ять дільників.

9. Знайти натуральне число із проміжку [a; b] із максимальною кількістю дільників.
10. Знайти суму дільників кожного з чисел з проміжку від 50 до 70.

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Богач І. В., Довгалець С. М., Дубовой В, М. Алгоритми розв’язання задач з програмування. Вінниця : ВНТУ, 2017. 119 с.
2. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123.
3. Дудзяний І. М. Програмування мовою С++. Частина 1 : Парадигма процедурного програмування : навчальний посібник. Львів : ЛНУ імені Івана Франка, 2013. 468 с.

Лабораторна робота 7. Розробка алгоритмів із використанням циклів з перед умовою та після умовою

Мета і зміст роботи: Розробка алгоритму, створення, налагодження та виконання програм, що використовують вкладені цикли з передумовою та післяумовою.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми із використанням циклів з передумовою та післяумовою;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Знайти добуток цифр заданого натурального числа (кількість цифр – довільна).

Алгоритм виконання:

1. Початок
2. Вхід: *N*
3. *S* = 0 // змінна для зберігання результату
4. поки N <> 0 виконувати
5. S = S + N%10;
6. N = N/10;
7. Вихід: S
8. Кінець

Реалізація на мові програмування:

```
#include <iostream>
using namespace std;
int main(){
   int N, S = 0;
   cout<<"Input N";
   cin>>N;
   while(N!=0){
      S+= N%10;
      N = N/10;
   }
   cout<<S;
```

```
return 0;
}
```

Результат роботи програми:

1) Input N 504

   0

2) Input N 3714

   84

3) Input N 98

   72

Завдання для індивідуального виконання:

1. Дано натуральне число. Чи правда, що всі його цифри розміщені у порядку зростання?
2. На скільки років необхідно покласти в банк суму в X грошових одиниць, щоб отримати суму в У грошових одиниць (У > X), якщо банк нараховує 14 % щорічних?
3. Перевірити, чи є задане число степенем 2. Наприклад, 32 – є, 10 – ні.
4. Перевірити, чи сума цифр заданого числа — парне число.
5. Дано натуральне число. Скільки разів у ньому зустрічається цифра *a*?
6. Знайти цифру із мінімальним значенням у записі числа. Наприклад, 391 – 1, 25 – 2, 10000 – 0.
7. Дано натуральне число n. «Викинути» із його запису цифри 0 та 5, залишивши усі інші. Наприклад, із числа 59015509 повинно стати 919.
8. Знайти добуток цифр заданого натурального числа (кількість цифр – довільна).
9. Першого дня спортсмен пробіг 7 км. Наступного дня — на 10% більшу відстань, ніж першого дня, третього дня він пробіг на 10% більше, ніж другого і т.д. На який день спортсмен вперше пробіжить відстань більшу, ніж 12 км?
10. Знайти цифру із максимальним значенням у записі числа. Наприклад, 391 – 9, 25 – 5, 1000 –1.

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Алгоритми, дані і структури : навч. посіб. / В. М. Ільман, О. П. Іванов, Л. О. Панік. Дніпро : Дніпропет. нац. ун-т залізн. трансп.ім. акад. В. Лазаряна, 2019. – 134 с.
2. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
3. Шаховська Н. Б. Алгоритми і структури даних. Навчальний посібник / Львів : Магнолія. – 216 с.
4. Онищенко В. В. Алгоритми та структури даних / В. В. Онищенко, Р. С. Коник. - К : 2017 - 66 с.

Лабораторна робота 8. Рекурсивні алгоритми

Мета і зміст роботи: Розробка рекурсивного алгоритму, створення, налагодження та виконання програм, що використовують рекурсивні алгоритми.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми, які використовують рекурсію;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Обчислити значення функції Аккермана для двох невід'ємних цілих чисел *n* та *m*, де:

$$A(n,m)=\begin{cases} m+1, якщо\ n=0 \\ A(n-1,1), якщо\ n\neq 0, m=0 \\ A(n-1,A(n,m-1)), якщо\ n>0, m>0 \end{cases}$$

Алгоритм виконання:

1. Початок Akkerm
2. Вхід: n, m
3. якщо *n = 0,* то повернути *m + 1* 7
4. інакше якщо m = 0, то обчислити значення Akkerm(n-1, 1)
5. інакше обчислити значення від Akkerm(n-1, Akkerm(n, m-1))
6. Вихід: Akkerm(n, m)
7. Кінець

Реалізація на мові програмування:

```
#include <iostream>
using namespace std;
unsigned int A(unsigned int n, unsigned int m){
   if (n==0){
     return m+1;}
   else{
      if ((n!=0) && (m==0)){
         return A(n-1,1);}
      else{
         return A(n-1,A(n,m-1));}
    }
}
```

```
int main(){
cout<< A(1,2)<<endl; // res = 4
cout<< A(0,1)<<endl;// res = 2
cout<< A(0,0)<<endl; // res = 1
cout<< A(2,2)<<endl; // res = 7
return 0;
}
```

Завдання для індивідуального виконання:

1. Дано натуральне число N. Перевірити, чи є дане число степенем 3. Для розв'язування даної задачі використати рекурсивну функцію.
2. Дано натуральне число N. Підрахувати суму його цифр. Для розв'язування даної задачі використати рекурсивну функцію.
3. Дано натуральне число N > 1. Перевірити, чи воно є простим.
4. Розкласти число на множники. Множники вивести на екран у порядку не спадання з врахуванням кратності.
5. Побудувати рекурсивний алгоритм піднесення числа x до степеня y.
6. Побудувати рекурсивний алгоритм множення числа x на число y.
7. Побудувати числову послідовність, в якій число k зустрічається рівно k разів. Наприклад: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, …
8. Реалізувати рекурсивну функцію знаходження площі прямокутника.
9. $$S(n,m)=\begin{cases}1, якщо\ n=m=1\\ S(n-1,m)+m, якщо\ n>1\\ S(n,m-1)+1, якщо\ m>1\end{cases}$$
10. Побудувати рекурсивний алгоритм для обчислення квадрату цілого числа, якщо відома залежність: $n^2 = (n\text{-}1)^2 + 2\cdot(n\text{-}1) + 1$.
11. Побудувати рекурсивний алгоритм для обчислити кількість комбінацій з n різних елементів по m. Кількість комбінацій визначається формулою
12. $$C_n^m=\begin{cases}1, якщо\ m=0, n>0\ або\ m=n\geq 0\\ 0, якщо\ m>n\geq 0\\ C_{n-1}^{m-1}+C_{n-1}^{m}, в\ інших\ випадках\end{cases}$$

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання

алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
2. Льовкін В. М. Методичні вказівки до виконання самостійної роботи студентів та розрахунково-графічних завдань з дисципліни “Алгоритмізація та програмування” для студентів спеціальності 122 “Комп’ютерні науки” (всіх форм навчання). Запоріжжя : ЗНТУ, 2017. 54 с.
3. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 2 : навчальний посібник. Львів : Видавництво «Новий Світ-2000», 2020. 320 с.
4. Ткачук В. М. Алгоритми та структура даних : навчальний посібник. Івано-Франківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
5. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. С++. Алгоритмізація та програмування: підручник. Одеса : Фенікс, 2019. 477 с.

Лабораторна робота 9. Рекурентні співвідношення

Мета і зміст роботи: Розробка алгоритму для обчислення значення рекурентного співвідношення, яке задається формулою.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми, які використовують рекурентні співвідношення;
- вміти виділяти рекурентну формулу для піднесення до степеня та обчислення факторіалу;
- вміти реалізовувати рекурсивну функцію, задану рекурентним співвідношенням;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Розробити алгоритм обчислення скінченної суми $\sum_{k=1}^{12} \frac{\cos^k(2x)}{k}$.

Значення параметра *x* отримується на вході алгоритму. Для частини загального члена необхідно обов'язково визначати рекурентну формулу (очевидно, що це стосуватиметься піднесення до степеня *k* чи обчислення факторіала)

Алгоритм виконання:

Для початку напишемо рекурентну формулу. Очевидно, що для $k = 1$ значення суми рівне $\cos\left(\frac{2x\pi}{180}\right)$. Якщо $k = 2$ маємо: $\cos^2\left(\frac{2x\pi}{180}\right)\cdot\frac{1}{2}$.

Рекурентна формула в такому випадку буде мати вигляд:

$$u_k = \begin{cases} \cos\left(\frac{2x\pi}{180}\right), k = 1 \\ u_{k-1}\cdot\cos\left(\frac{2x\pi}{180}\right)\cdot\frac{1}{k}, k = \overline{2,11} \end{cases}.$$

Тоді функція для обчислення значення заданої скінченної суму матиме вигляд:

1. Початок
2. Вхід: x
3. u – член рекурентного співвідношення, s – значення суми
4. u=cos(2*x), s=u;
5. для k від 1 до 12; виконувати

6. u*=cos(2*x*PI/180), +=u/k
7. Вихід: s
8. Кінець

Реалізація на мові програмування:

```
#include <iostream>
#include <cmath>
using namespace std;
const double PI=3.14159265;
int main(){
float x, s, u;
cout<<"x=";
cin>>x;
int k=1;
for ( u=(float)cos(2*x*PI/180), s=u; k<=12; k++,
u*=cos(2*x*PI/180), s+=u/k){
cout<<u<<" ";
cout<<s<<" "<<endl;}
cout<<s;
return 0;
}
```

Завдання для індивідуального виконання:

Розробити алгоритм обчислення скінченної суми. Значення параметра x отримується на вході алгоритму. Для частини загального члена необхідно обов'язково визначати рекурентну формулу (очевидно, що це стосуватиметься піднесення до степеня k чи обчислення факторіала.

1. $\sum_{k=1}^{11} \frac{kx^{k-1}}{4k^2+1}$

2. $\sum_{k=1}^{12} x^k \cos^k\left(\frac{k}{4}\right)$

3. $\sum_{k=3}^{14} \frac{k^2 \cos^k x}{\left(2k^2-1\right)}$

4. $\sum_{k=2}^{11} \frac{k^2 \sin^k x}{\left(2k^2-1\right)}$

5. $\sum_{k=1}^{11} \frac{(-1)^k x^{k-1}}{(k-1)!}$

6. $\sum_{k=1}^{8} \frac{k^2 x^k}{(2k)!}$

7. $\sum_{k=3}^{14} \frac{(-1)^k x^k}{k^2}$

8. $\sum_{k=1}^{11} \frac{kx^k}{\left(4k^2-1\right)}$

9. $\sum_{k=1}^{9} \frac{\sin(2kx)}{(2k)!}$

10. $\sum_{k=1}^{10} k(2k+1)x^k$

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання

алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
2. Льовкін В. М. Методичні вказівки до виконання самостійної роботи студентів та розрахунково-графічних завдань з дисципліни “Алгоритмізація та програмування” для студентів спеціальності 122 “Комп’ютерні науки” (всіх форм навчання). Запоріжжя : ЗНТУ, 2017. 54 с.
3. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 2 : навчальний посібник. Львів : Видавництво «Новий Світ-2000», 2020. 320 с.
4. Ткачук В. М. Алгоритми та структура даних : навчальний посібник. Івано-Франківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
5. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. С++. Алгоритмізація та програмування: підручник. Одеса : Фенікс, 2019. 477 с.

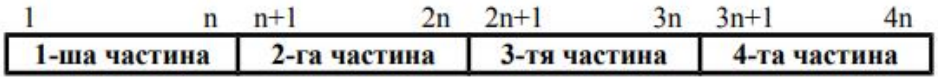
Лабораторна робота 10. Масиви. Алгоритми зсуву масиву

Мета і зміст роботи: Розробка алгоритмів для перетворення масивів “на тому ж місці”, створення, налагодження та виконання програм, що використовують алгоритми обробки масивів.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми, які модифікують одновимірні масиви;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблений алгоритм на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Задано натуральне число *n* та одновимірний масив чисел A[4*n*], який складається із 4 частин по *n* елементів кожна.



Перетворити масив “на тому ж місці” (тобто результуючий масив повинен бути на тому ж місці, що і вхідний), переставивши елементи у порядку, вказаному у вашому варіанті.

Завдання розв’язати двома способами:

1) для виконання перетворення використати додатковий масив B розмірністю не більше 4n. Кращим варіантом рішення буде використання масиву розмірністю n.

2) для виконання перетворення використати лише одну проміжну змінну.

Позначення:

1. Цифри 1, 2, 3, 4 позначають номер частини початкового масиву A. У варіантах завдань ці цифри стоять на тих місцях, на які повинна бути переставлена частина масиву, позначена певним номером.

2. Стрілка вправо означає, що порядок взаємного розташування відповідної частими масиву A повинен бути збереженим таким самим, як у початковому масиві.

3. Стрілка вправо означає, що порядок взаємного розташування відповідної частими масиву A повинен бути взаємно оберненим до порядку розміщення елементів у початковому масиві.

Реалізація на мові програмування варіанту №1 із завдань для індивідуального виконання:

Варіант із використанням додаткового масиву розмірності n.

```
#include <iostream>
using namespace std;
int main(){
const int N=4;
int A[4*N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16}, B [N];
for (int i=0; i<N; i++){ // 3 та 4
B[i] = A[3*N+i];
A[3*N+i] = A[2*N+i];
A[2*N+i] = B[i];
}
for (int i=0; i<N; i++){ // 2 та 3
B[i] = A[2*N+i];
A[2*N+i] = A[i+N];
A[N+i] = B[i];}
for (int i=0; i<N; i++){ // переписати 2
B[i]=A[2*N-1-i];}
for (int i=0; i<N; i++){
A[1*N + i] = B[i];}
for (int i=0; i<N; i++){// переписати 4
B[i]=A[4*N-1-i];}
for (int i=0; i<N; i++){
A[3*N + i] = B[i];
}
for (int i=0; i<4*N; i++){
cout<<A[i]<<" ";}
return 0;
}
```

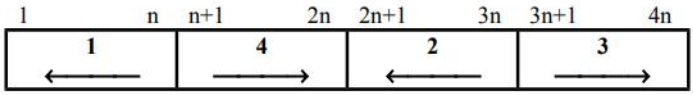
Варіант із використанням однієї проміжної змінної.

```
#include <iostream>
using namespace std;
int main(){
const int N=4;
int A[4*N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16};
int c;
for (int i=0; i<N; i++){ // 3 та 4
c = A[3*N+i];
A[3*N+i] = A[2*N+i];
A[2*N+i] = c;}
for (int i=0; i<N; i++){ // 2 та 3
```

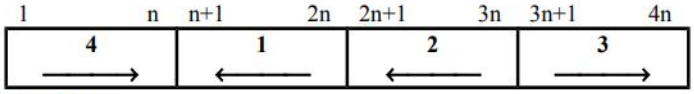
```
c = A[2*N+i];
A[2*N+i] = A[i+N];
A[N+i] = c;}
for (int i=0; i<2; i++){ // переписати 2
c = A[2*N-1-i];
A[2*N-1-i] = A[1*N + i] ;
A[1*N + i]= c;}
for (int i=0; i<2; i++){// переписати 4
c=A[4*N-1-i];
A[4*N-1-i] = A[3*N + i];
A[3*N + i] = c;}
for (int i=0; i<4*N; i++){
cout<<A[i]<<" ";}
return 0;
}
```

Завдання для індивідуального виконання:

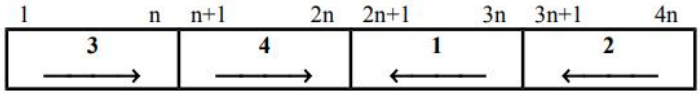
Варіант 1



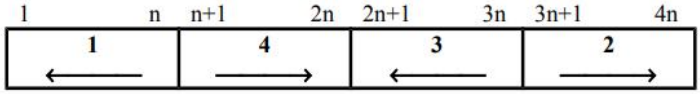
Варіант 2



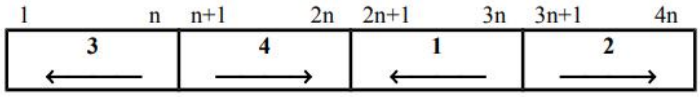
Варіант 3



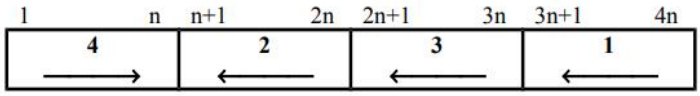
Варіант 4



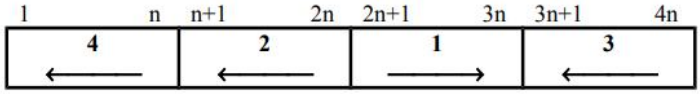
Варіант 5



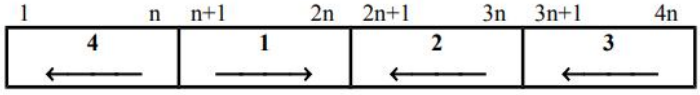
Варіант 6



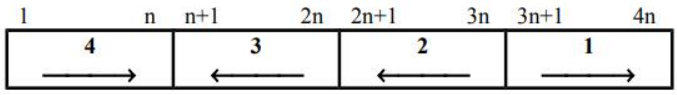
Варіант 7



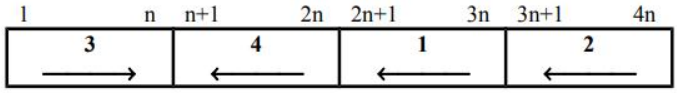
Варіант 8



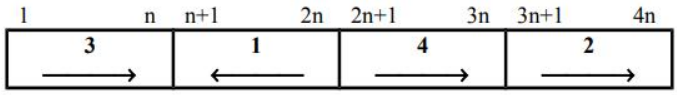
Варіант 9



Варіант 10



Варіант 11



Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
2. Льовкін В. М. Методичні вказівки до виконання самостійної роботи студентів та розрахунково-графічних завдань з дисципліни “Алгоритмізація та програмування” для студентів спеціальності 122 “Комп’ютерні науки” (всіх форм навчання). Запоріжжя : ЗНТУ, 2017. 54 с.
3. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 2 : навчальний посібник. Львів : Видавництво «Новий Світ-2000», 2020. 320 с.
4. Ткачук В. М. Алгоритми та структура даних : навчальний посібник. Івано-Франківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
5. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. С++. Алгоритмізація та програмування: підручник. Одеса : Фенікс, 2019. 477 с.

Лабораторна робота 11. Алгоритм лінійного пошуку. Лінійний пошук з бар'єром

Мета і зміст роботи: Застосування алгоритму лінійного (прямого) пошуку та лінійного (прямого) пошуку з бар'єром, створення, налагодження та виконання програм, що використовують ці алгоритми.

В результаті виконання роботи необхідно:

- вміти застосовувати алгоритм лінійного пошуку при розв'язанні задач;
- вміти застосовувати алгоритм лінійного пошуку з бар'єром при розв'язанні задач;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблені алгоритми на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Задано двовимірний масив цілих чисел A[n, n]. На головній діагоналі знайти перший нульовий та останній від'ємний елементи.

Алгоритм виконання:

1. Початок
2. Вхід: розмірність масиву n, двовимірний масив A
3. проміжні змінні i, j, numZero, numNeg
4. для i від 0 до n виконати
5. для j від 0 до n виконати
6. A[i][j] = 5 - rand()%11
7. кінець циклу по j
8. кінець циклу по i
9. для i від 0 до n виконати
10. якщо A[i][i] = 0 то numZero = i вихід
11. для i від 0 до n виконати
12. якщо A[i][i] < 0 то numNeg = i
13. Вихід: numZero, numNeg
14. Кінець

Реалізація на мові програмування:

```
#include <iostream>
```

```
using namespace std;

int main()
{
const int n=5;
int A[n][n];
for (int i=0; i<n; i++){
for (int j=0; j<n; j++){
A[i][j] = 5 - rand()%11;
cout<<A[i][j]<<" ";
}
cout<<endl;
}
int numZero;
for (int i=0; i<n; i++){
if (A[i][i]==0){
numZero = i;
break;
}
}
int numLessZero;
for (int i=0; i<n; i++){
if (A[i][i]<0){
numLessZero = i;
}
}
A[numZero][numZero] = A[numLessZero][numLessZero];
A[numLessZero][numLessZero] = 0;
for (int i=0; i<n; i++){
for (int j=0; j<n; j++){
cout<<A[i][j]<<" ";
}
cout<<endl;
}
return 0;
}
```

Завдання для індивідуального виконання:

1. Задано двовимірний масив цілих чисел A[n, n]. На головній діагоналі знайти перший додатній та останній від'ємний елементи та поміняти їх місцями.
2. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по рядках зайти перший від'ємний елемент та його координати (номер рядка та номер стовпця).
3. Задано двовимірний масив цілих чисел A[n, n]. На бічній діагоналі знайти перший додатній та останній від'ємний елементи та поміняти їх місцями.

4. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по стовпцях зайти перший від'ємний елемент та його координати (номер рядка та номер стовпця).
5. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по рядках зайти перший додатний елемент та його координати (номер рядка та номер стовпця).
6. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по рядках зайти останній додатний елемент та його координати (номер рядка та номер стовпця).
7. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по рядках зайти останній від'ємний елемент та його координати (номер рядка та номер стовпця).
8. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по стовпцях зайти останній додатний елемент та його координати (номер рядка та номер стовпця).
9. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по стовпцях зайти останній від'ємний елемент та його координати (номер рядка та номер стовпця).
10. Задано двовимірний масив цілих чисел A[n, n]. При обході масиву по рядках зайти перший нульовий елемент та його координати (номер рядка та номер стовпця).

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

1. Махровська Н.А., Погромська Г. С. Алгоритми і структури даних: навчально-методичний посібник. Миколаїв : МНУ ім. В.О. Сухомлинського, 2019. 279 с.
2. Прийма С.М. Теорія алгоритмів: навчальний посібник. Мелітополь: ФОП Однорог Т. В., 2018. 116 с.

Лабораторна робота 12. Алгоритм бінарного пошуку. Застосування бінарного пошуку

Мета і зміст роботи: Знати алгоритм бінарного пошуку, вміти реалізовувати алгоритм бінарного пошуку на мові програмування, вміти застосовувати алгоритм бінарного пошуку для знаходження елемента у рядках та стовпцях матриці.

В результаті виконання роботи необхідно:

- знати алгоритм бінарного пошуку;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати алгоритм бінарного пошуку на мові програмування;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Задано матрицю A[n, n]. На головній діагоналі матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цієї діагоналі впорядковані у порядку незростання.

Алгоритм:

1. Початок
2. Вхід: двовимірний масив M, розмірність масиву n, шукане значення key
3. $L = 0$; $R = N$-1
4. поки $L < R${
5. $m = (L+R) / 2$
6. якщо $a[m] < Key$
7. то $L = m+1$
8. інакше $R = m$
9. }
10. якщо $a[R] = Key$
11. то елемент має номер R
12. інакше елемент відсутній
13. Вихід: індекс R шукано значення key
14. Кінець

Завдання:

```
#include <iostream>
```

```
using namespace std;
int main(){
int const N = 5;
int B[N], M[N][N];
for (int i = 0; i < N; i++){
for (int j = 0; j < N; j++){
M[i][j] = rand()%21;
cout << M[i][j] << " ";
}
cout << endl;
}
for (int i = 0; i < N; i++){
for (int j = 0; j < N; j++){
if (i + j == 4){
B[i] = M[i][j];
cout << B[i] << " ";}
}
}
int l = 0, r = N - 1, x, m, max, maxi;
cin >> x;
for (int i = 0; i < N; i++){
if (B[i] == x){
max = B[i];
maxi = i;
cout << maxi << endl;}
}
while (l <= r){
m = (l + r) / 2;
if (B[m] == x){
cout << "Дане число присутнє на діагоналі" << endl;
break;}
else{
if (B[m] < x){
r = m - 1;}
else{
l = m + 1;}
}
}
if (l > r){
cout << " Дане число відсутнє на діагоналі ";}
}
```

Завдання для індивідуального виконання:

1. Задано матрицю A[m, n]. Окремо у кожному рядку матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи кожного рядка окремо впорядковані у порядку неспадання.
2. Задано матрицю A[m, n]. Окремо у кожному стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму

двійкового (бінарного) пошуку, якщо елементи кожного стовпця окремо впорядковані у порядку неспадання.

3. Задано матрицю A[n, n]. На головній діагоналі матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи на головній діагоналі впорядковані у порядку неспадання.
4. Задано матрицю A[m, n]. У першому рядку та останньому стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.
5. Задано матрицю A[m, n]. В останньому рядку та першому стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.
6. Задано матрицю A[m, n]. В останньому рядку та останньому стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.
7. Задано матрицю A[m, n]. В першому рядку та останньому стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.
8. Задано матрицю A[m, n]. В другому рядку та (n-2)-му стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.
9. Задано матрицю A[m, n]. В (m-2)-му рядку та другому стовпцеві матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи цих рядка та стовпця впорядковані у порядку неспадання.

10. Задано матрицю A[m, n]. Окремо у кожному рядку матриці визначити присутність цілого числа x та його індекс шляхом застосування алгоритму двійкового (бінарного) пошуку, якщо елементи кожного рядка окремо впорядковані у порядку незростання.

Звіт до лабораторної роботи повинен містити розроблений алгоритм, виконуваний код програми. Здобувач повинен пояснювати хід виконання алгоритму, програми відповідати на питання щодо роботи алгоритму, програми, теми лабораторної роботи.

Рекомендовані джерела

3. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
4. Мелешко Є. В., Якименко М. С., Поліщук Л. І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. Кропивницький: Видавець Лисенко В. Ф., 2019. 156 с.
5. Махровська Н.А., Погромська Г. С. Алгоритми і структури даних: навчально-методичний посібник. Миколаїв : МНУ ім. В.О. Сухомлинського, 2019. 279 с.
6. Прийма С.М. Теорія алгоритмів: навчальний посібник. Мелітополь: ФОП Однорог Т. В., 2018. 116 с.
7. Сергієнко А. М., Марченко О. І. Конспект лекцій по курсу “Алгоритми і структури даних” для напряму підготовки 123 Комп’ютерна інженерія. К : Національний технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», 2017. 74 с.

Лабораторна робота 13. Алгоритми сортування числових даних

Мета і зміст роботи: Ознайомитись із основними алгоритмами сортування даних, зокрема, алгоритмом сортування вставки, вибору, злиття, Шелла, вміти використовувати алгоритми сортування для впорядкування стовпців та рядків у двовимірному масиві.

В результаті виконання роботи необхідно:

- вміти розробляти алгоритми сортування методом вставки, вибору та злиття;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблені алгоритми сортування на мові програмування;
- вміти оформляти алгоритми сортування у вигляді функцій;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Завдання: Реалізувати алгоритм обмінного сортування вибіркою.

Алгоритм:

1. Початок
2. Вхід: масив a[n]
3. Знаходимо мінімальний елемент, переставляємо його (робимо взаємозаміну) з першим.
4. Знаходимо мінімальний елемент в залишеному невпорядкованому масиві a[2] .. a[n], переставляємо його із a[2]
5. . . . .
6. n-2) Знаходимо мінімальний елемент в масиві (a[n-2], a[n-1], a[n]), переставляємо його із a[n-2].
7. n-1) Впорядковуємо залишену пару a[n] і a[n-1].
8. Вихід: впорядкований масив a[n]
9. Кінець

Реалізація на мові програмування:

```
#include <iostream>
using namespace std;
void swap(int *xp, int *yp){
int temp = *xp;
```

```
*xp = *yp;
*yp = temp;
}
void selectionSort(int arr[], int n){
int i, j, min_idx;
for (i = 0; i < n-1; i++){
min_idx = i;
for (j = i+1; j < n; j++)
if (arr[j] < arr[min_idx])
min_idx = j;
swap(&arr[min_idx], &arr[i]);}
}
void printArray(int arr[], int size){
int i;
for (i=0; i < size; i++)
cout << arr[i] << " ";
cout << endl;
}
int main(){
int arr[] = {64, 25, 12, 22, 11};
int n = sizeof(arr)/sizeof(arr[0]);
selectionSort(arr, n);
cout << "Sorted array: \n";
printArray(arr, n);
return 0;
}
```

Завдання для індивідуального виконання:

1. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи непарних рядків масиву методом вставки за зростанням.
2. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи парних рядків масиву методом вставки за зростанням.
3. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи непарних стовпців масиву методом вибірки за спаданням.
4. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи парних стовпців масиву методом вибірки за спаданням.
5. Задано двовимірний масив цілих чисел A[m, m]. Відсортувати елементи головної діагоналі масиву методом Шелла за зростанням.
6. Задано двовимірний масив цілих чисел A[m, m]. Відсортувати елементи бічної діагоналі масиву методом Шелла за спаданням.
7. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи кожного стовпця масиву методом шейкерного сортування за спаданням.

8. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи кожного стовпця масиву методом шейкерного сортування за зростанням.
9. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи кожного рядка масиву методом швидкого сортування Хоара за спаданням.
10. Задано двовимірний масив цілих чисел A[m, n]. Відсортувати елементи кожного рядка масиву методом швидкого сортування Хоара за зростанням.

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
2. Мелешко Є. В., Якименко М. С., Поліщук Л. І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. Кропивницький: Видавець Лисенко В. Ф., 2019. 156 с.
3. Махровська Н.А., Погромська Г. С. Алгоритми і структури даних: навчально-методичний посібник. Миколаїв : МНУ ім. В.О. Сухомлинського, 2019. 279 с.
4. Прийма С.М. Теорія алгоритмів: навчальний посібник. Мелітополь: ФОП Однорог Т. В., 2018. 116 с.
5. Сергієнко А. М., Марченко О. І. Конспект лекцій по курсу “Алгоритми і структури даних” для напряму підготовки 123 Комп’ютерна інженерія. К : Національний технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», 2017. 74 с.

Лабораторна робота 14. Порівняльний аналіз алгоритмів сортування

Мета і зміст роботи: Ознайомлення із параметрами, за якими проводиться оцінка швидкості виконання алгоритмів сортування, вимірювання часу виконання алгоритмів сортування, вміння адаптувати готові програмні розробки для поставленої задачі.

В результаті виконання роботи необхідно:

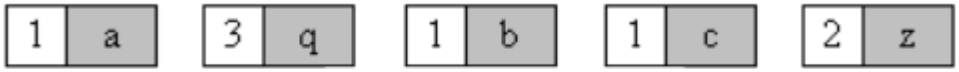
- вміти розробляти алгоритми сортування методом вставки, вибору та злиття;
- знати, як працює алгоритм сортування методом Шелла;
- вміти обирати оптимальний тип для представлення вхідних та вихідних даних;
- вміти реалізовувати розроблені алгоритми сортування на мові програмування;
- вміти адаптувати готову програмну реалізацію алгоритму сортування Шелла для власної програмної розробки;
- вміти виконувати тестування програми;
- вміти пояснювати хід виконання алгоритму та принцип роботи програми.

Параметри, за якими проводиться оцінка алгоритмів:

1. Час впорядкування–основний параметр, що характеризує швидкодію алгоритму.

2. Пам'ять–ряд алгоритмів вимагає виділення додаткової пам'яті під тимчасове зберігання даних. При оцінці використовуваної пам'яті не враховуватиметься місце, яке займає вихідний масив і незалежні від вхідної послідовності витрати, наприклад, на зберігання коду програми.

3. Стійкість–стійка впорядкуванняне змінює взаємного розташування рівних елементів. Така властивість може бути дуже корисним, якщо вони складаються з декількох полів, як на рис. 1, а впорядкуваннявідбувається по одному з них, наприклад, по x.



Вихідні дані

Приклад роботи стійкого впорядкування

Взаємне розташування рівних елементів з ключем 1 і додатковими полями "a", "b", "c" залишилося колишнім: елемент з полем "a", потім з "b", потім з "c".

Приклад роботи нестійкого впорядкування

Взаємне розташування рівних елементів з ключем 1 і додатковими полями "a", "b", "c" змінилося.

4. Природність поведінки–ефективність методу при обробці вже відсортованих, або частково відсортованих даних. Алгоритм поводиться природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

Для того, щоб знайти час виконання програми, потрібно використати функцію clock(). Прототип функції clock()знаходиться в заголовковому файлі <ctime>, який потрібно підключити на початку програми. Функція clock() повертає значення часу в мілісекундах (1с = 1000млс). Причому відлік часу починається з момента запуску програми. Якщо потрібно виміряти час виконання всієї програми, то в кінці програми, перед оператором return 0; потрібно запустити функцію clock(), яка покаже робочий час. Для пошуку часу виконання фрагмента коду потрібно знайти різницю між початковим та кінцевим часом.

Приклад 1

```
// Як знайти час виконання фрагмента програми?
 // заголовковий файл з прототипом потрібної функції clock()
#include <ctime>
//          ...
    unsigned int start_time =  clock(); // початковий час
// тут повинен бути фрагмент програми, час виконання якого потрібно обчислити
    unsigned int end_time = clock(); // кінцевий час
    unsigned int search_time = end_time - start_time; // шуканий час
```

Приклад 2

```
// Як знайти час роботи програми?
 #include <ctime>
//          ...
// тут повинен бути код програми, час роботи якого потрібно виміряти
    unsigned int end_time = clock(); // час роботи програми
```

Завдання: Провести порівняльний аналіз алгоритмів сортування, запропонованих у таблиці. Час виконання виміряти для різної кількості елементів одновимірного масиву –10*N, 100*N, 10000*N. Дані записати в таблицю. (Для вимірювання часу виконання використати засоби мови програмування, вказати тип операційну систему та тип процесора).

Алгоритм	Час (в мілісекундах)		
	110N	1100N	10000N
Сортування вибіркою* (Selection Sort)			
Сортування вставкою (Insertion Sort)			
Сортування злиттям (Merge Sort)			
Сортування Шелла (Shell Sort)			

* (вказати, чи це обмінне сортування чи просте)

Рекомендовані джерела

1. Гришанович Т. О. Алгоритми і структури даних [Електронний ресурс] : електронний курс навчальної дисципліни, затверджений НМР ВНУ імені Лесі Українки, протокол № 4 від 16.12.2020 / Тетяна Гришанович. – ВНУ ім. Лесі Українки, 2020. // Режим доступу : http://cs.vnu.edu.ua/moodle/course/view.php?id=123
2. Мелешко Є. В., Якименко М. С., Поліщук Л. І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. Кропивницький: Видавець Лисенко В. Ф., 2019. 156 с.
3. Махровська Н.А., Погромська Г. С. Алгоритми і структури даних: навчально-методичний посібник. Миколаїв : МНУ ім. В.О. Сухомлинського, 2019. 279 с.
4. Прийма С.М. Теорія алгоритмів: навчальний посібник. Мелітополь: ФОП Однорог Т. В., 2018. 116 с.
5. Сергієнко А. М., Марченко О. І. Конспект лекцій по курсу “Алгоритми і структури даних” для напряму підготовки 123 Комп’ютерна інженерія. К : Національний технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», 2017. 74 с.

УДК 004.421:004.422.63(076)

Електронне мережне навчальне видання

Т. О. Гришанович

ЛАБОРАТОРНИЙ ПРАКТИКУМ ІЗ ДИСЦИПЛІНИ “АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ”

для студентів спеціальності 122 Комп’ютерні науки
першого (бакалаврського) рівня

Друкується в авторській редакції