Лабораторна робота №3

Побудова за допомогою програми Rational Rose UML діаграми діяльності автоматизованої інформаційної системи.

***Мета роботи:*** *засвоїти методику та виробити практичні в побудові за допомогою програми Rational Rose UML діаграми класів автоматизованої інформаційної системи.*

Теоретичні відомості

UML діаграми діяльності

Залежно від ступеня деталізації, діаграми діяльностей використовуються на різних етапах розробки. На етапі аналізу вимог і уточнення специфікацій діаграми діяльностей дозволяють конкретизувати основні функції розроблюваного програмного забезпечення.

Під діяльністю в цьому випадку розуміють завдання (операцію), яке необхідно виконати вручну або за допомогою засобів автоматизації. Кожному варіанту використання відповідає своя послідовність завдань.

Діаграми діяльності є представленням алгоритмів якихось дій, що виконуються в системі.

Алгоритм – це послідовність певних дій або елементарних операцій, виконання яких приводить до здобуття бажаного результату.

Можна побудувати декілька діаграм діяльності для однієї і тієї ж системи, причому кожна з них фокусуватиметься на різних аспектах системи, показуватиме різні дії, що виконуються усередині неї.

Розглянемо приклад діаграми діяльності на щоранкових діях людини, що зображено на рис. 1.



Рис. 1. UML-діаграма діяльності на прикладі щоранкових дій людини.

На діаграмі діяльності, що розглянута вище, дії зображені заокругленими прямокутниками. Ромб на діаграмі діяльності – це символ прийняття рішень з позначкою умов біля переходів. Початок діаграми зображується замальованим колом (рис. 2. а), а кінець діаграми зображений у вигляді символу, який нагадує котячі очі (рис. 2. б).



Рис. 5.14. а - початок діаграми; б- кінець діаграми.

Товсті лінії перед та після дій Душ та Спів означають розпаралелювання, а потім знов злиття потоків керування. Тобто дії Душ та Спів виконуються одночасно.

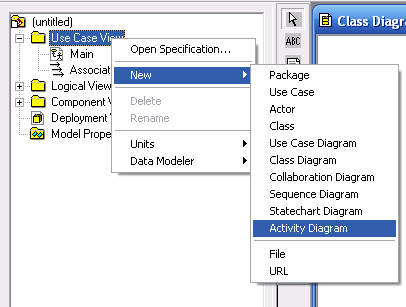
На діаграмі діяльності можна не лише показати паралельне виконання дії, але і вказати стани об'єктів, також є можливість показувати розподіл ролей і потік об’єкта.

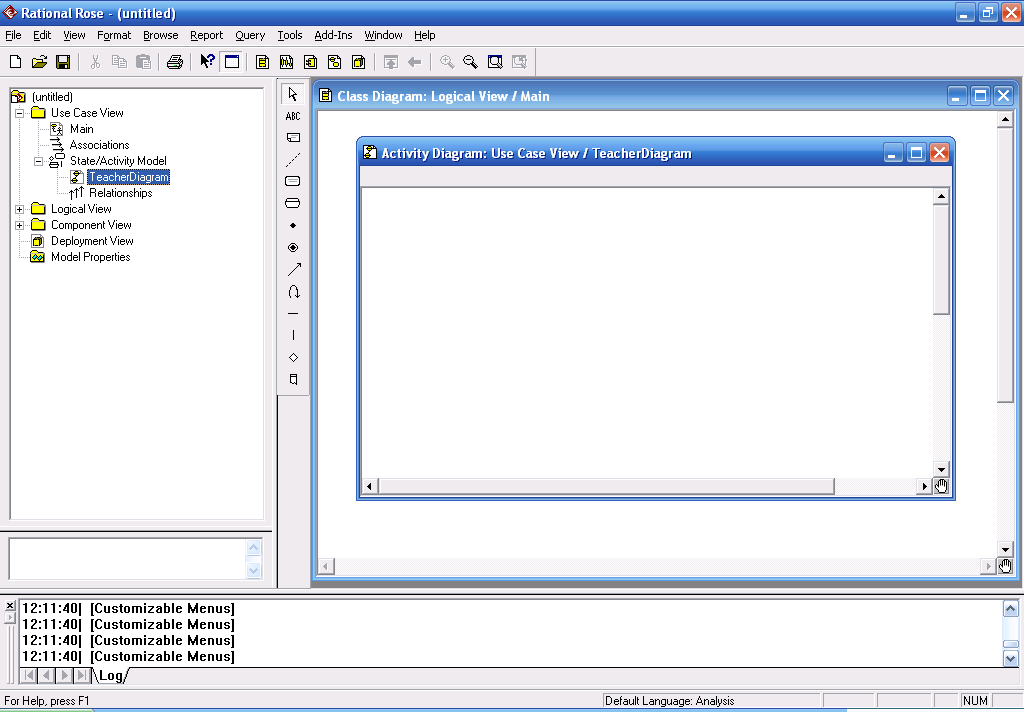
Суть потоку об’єкта полягає в тому, що на діаграмі діяльності можна відобразити об'єкти, що відносяться до діяльності. За допомогою символу залежності (пунктирна стрілка) ці об'єкти можна співвіднести з тією діяльністю або переходом, де вони створюються, змінюються або знищуються. Уявимо таку ситуацію з повсякденного життя: ви приходите в який-небудь фастфуд і замовляєте гамбургер з колою. Під час приготування сніданку кухар створює новий об'єкт - гамбургер. Поки ви нетерпляче випиваєте колу, офіціант переміщає цей об'єкт (подає ваше замовлення). Природно, під час сніданку ви знищуєте цей об'єкт. От як це виглядає на діаграмі що зображена на рисунку 3.



Рис. 3. UML-діаграма діяльності з розподілом ролей та з потоком об’єкту.

Почати побудову діаграми діяльності для обраного елемента моделі або модельованої системи в можна ( Рис.4. ) виділивши логічне подання (Logical View) або подання варіантів використання (Use Case View ) в браузері проекту і виконати операцію контекстного меню: New/Activity Diagram  ( Новая/Діаграмма діяльності).



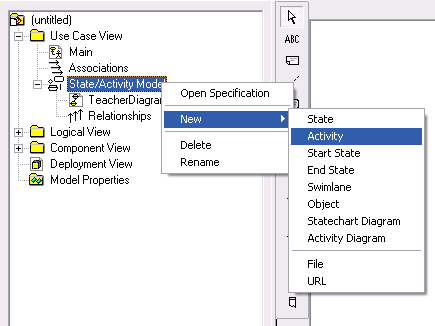


В результаті виконання цих дій з'являється нове вікно з чистим робочим листом діаграми діяльності і спеціальна панель інструментів , що містить кнопки із зображенням графічних елементів , необхідних для розробки діаграми діяльності ( Табл. 1). Призначення окремих кнопок панелі можна дізнатися з випливаючих підказок.

|  |  |  |
| --- | --- | --- |
| Графічне зображення | Випливаюча підказка | Призначення кнопки |
|  | Selection Tool | Перетворює зображення курсору у форму стрілки для подальшого виділення елементів на діаграмі |
|  | Text Box | Додає на діаграму текстову область |
|  | Note | Додає на діаграму примітка |
|  | Anchor Note to Item | Додає на діаграму зв'язок примітки з відповідним графічним елементом діаграми |
|  | State | Додає на діаграму стан |
|  | Activity | Додає на діаграму діяльність |
|  | Start State | Додає на діаграму початковий стан |
|  | End State | Додає на діаграму кінцевий стан |
|  | State Transition | Додає на діаграму перехід |
|  | Transition to Self | Додає на діаграму рефлексивний перехід |
|  | Horizontal Synchronization | Додає на діаграму горизонтально розташований символ синхронізації |
|  | Vertical Synchronization | Додає на діаграму вертикально розташований символ синхронізації |
|  | Decision | Додає на діаграму символ прийняття рішення для альтернативних переходів |
|  | Swimlane | Додає на діаграму доріжку |
|  | Object | Додає на діаграму об'єкт (за замовчуванням відсутній) |
|  | Object Flow | Додає на діаграму стрілку потоку об'єктів (за замовчуванням відсутній) |
|  | Business Activity | Додає на діаграму бізнес-діяльність (за замовчуванням відсутній) |
|  | Business Transaction | Додає на діаграму бізнес-транзакцію (за замовчуванням відсутній) |

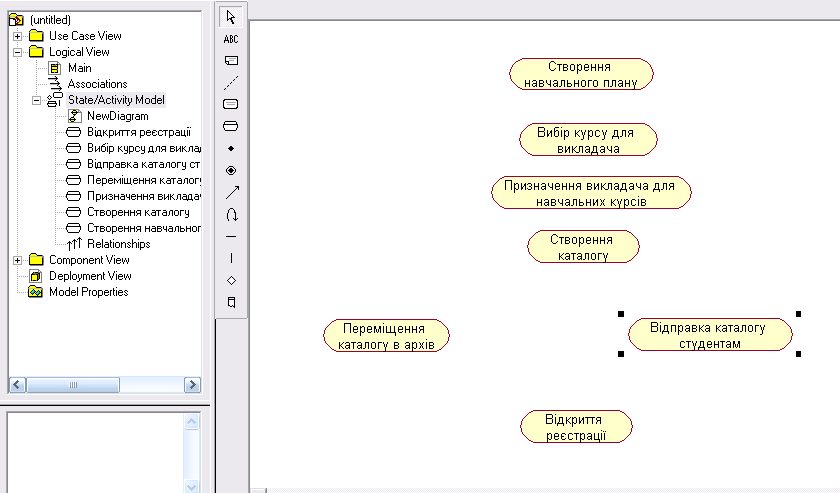
Як видно з цієї таблиці, за замовчуванням на панелі інструментів відсутні деякі графічні елементи, а саме - кнопки з піктограмами об'єкта й потоку об'єктів. При необхідності їх можна додати на спеціальну панель діаграми діяльності стандартним способом.

Для додавання діяльності на діаграму діяльності потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням піктограми діяльності на спеціальній панелі інструментів, відпустити ліву кнопку миші і клацнути лівою кнопкою миші на вільному місці робочого листа діаграми. На діаграмі з'явиться зображення діяльності з маркерами зміни його геометричних розмірів і запропонованим середовищем ім'ям за замовчуванням, яке розробнику слід змінити. Додати діяльність на діаграму можна також за допомогою операції головного меню: Tools/Create/Activity або за допомогою операції контекстного меню: New/Activity , попередньо виділивши діаграму діяльності в браузері проекту ( Рис.1.3 ) .

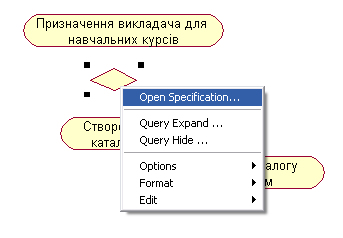


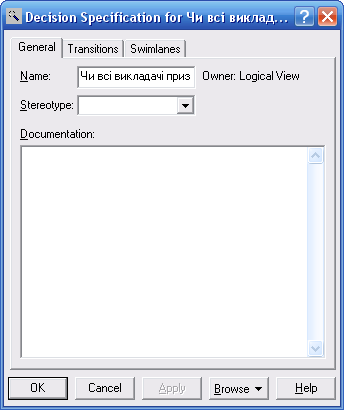
В результаті цих дій на діаграмі з'явиться зображення діяльності з ім'ям NewActivity , запропоноване програмою за замовчуванням.

Розглянемо діаграму діяльності на основі АІС реєстрації навчальних курсів.



Тепер додамо на діаграму елемент прийняття рішення (розгалуження) для альтернативних переходів (рішення), "що задає питання": "Чи всі викладачі призначені?"





Додаємо на діаграму горизонтально розташований символ синхронізації (горизонтальна синхронізація) , перетягуючи його зі спеціальної панелі інструментів на область діаграми діяльності .

Для додавання переходу між двома діяльностями потрібно за допомогою лівої кнопки миші натиснути кнопку із зображенням переходу на спеціальній панелі інструментів , відпустити ліву кнопку миші , клацнути лівою кнопкою миші на зображенні вихідної діяльності на діаграмі і відпустити її на зображенні цільової діяльності .

Додамо на діаграму всі можливі переходи та отримаємо:



Розглянемо приклад створення UML діаграми по програмі на мові програмування С++.

Умова програми:

Для сукупності чисел (0.1,-25.2,0,888.35,-424.48,0,15,-7,-195,2,0,0.04,-15.2;57.6). Знайти номер елемента з мінімальним значенням.

Код програми:

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int i, k;

float x[]={0.1, -25.2, 0.0, 888.35, -424.48, 0.0, 15.0, -7.0, -195.0, 2.0, 0.0, 0.04, -15.2, 57.6};

float min;

clrscr();

printf("Utvorenuy masuv\n");

for(i=0;i<14;i++)

printf("x[%d]= %3.2f\n", i, x[i]);

min=x[0];

k=0;

for(i=0;i<14;i++)

{

if(min>x[i])

{

min=x[i];

k=i;

}

}

printf("Min element masuvy = %3.2f\n", min);

printf("i= %d", k);

getch();

return 0;

}

UML діаграма:



Завдання

Побудувати за допомогою програми Rational Rose UML діаграму діяльності згідно варіанту.

Варіанти

|  |  |  |
| --- | --- | --- |
| **1** | Для сукупності чисел (0.1, -25.2, 0, 888.35,  -424.48, 0, 15, -7, -195, 2, 0, 0.04, -15.2; 57.6). Обчислити добуток елементів масиву, менших за 4,4. | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[]={0.1, -25.2, 0.1, 888.35, -424.48, 0.1, 15.0, -7.0, -195.0, 2.0, 0.1, 0.04, -15.2, 57.6};  float d;  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<14;i++)  printf("x[%d]= %3.2f\n", i, x[i]);  d=1;  for(i=0;i<14;i++)  {  if(x[i]<4.4)  d=d\*x[i];  }  printf("Dobutok elementiv masuvy < 4.4 = %3.2f", d);  getch();  return 0;  } |
| **2** | Для сукупності чисел (0.1, -25.2, 0, 888.35,  -424.48, 0, 15, -7, -195, 2, 0, 0.04, -15.2; 57.6. Знайти мінімальний елемент масиву серед від’ємних. | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[]={0.1, -25.2, 0.0, 888.35, -424.48, 0.0, 15.0, -7.0, -195.0, 2.0, 0.0, 0.04, -15.2, 57.6};  float min;  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<14;i++)  printf("x[%d]= %3.2f\n", i, x[i]);  min=x[0];  for(i=0;i<14;i++)  {  if(x[i]<0)  {  if(min>x[i])  min=x[i];  }  }  printf("Min element masuvy sered - = %3.2f", min);  getch();  return 0;  } |
| **3** | Для сукупності чисел (0.1, -25.2, 0, 888.35,  -424.48, 0, 15, -7, -195, 2, 0, 0.04, -15.2; 57.6. Обчислити суму останніх шести елементів масиву | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[]={0.1, -25.2, 0.0, 888.35, -424.48, 0.0, 15.0, -7.0, -195.0, 2.0, 0.0, 0.04, -15.2, 57.6};  float S;  clrscr();  printf("Utvoreni masuvu\n");  for(i=0;i<=13;i++)  printf("x[%d]= %3.2f\n", i, x[i]);  S=0;  for(i=8;i<=13;i++)  S=S+x[i];  printf("Suma ostannih 6 elementiv masuvy = %3.2f", S);  getch();  return 0;  } |
| **4** | Утворити масив i вивести на друк елементи масиву. *xi = ei +2,5i.*  Утворити вектор *, де 1<=i<=10, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[10], y[10];  float max;  clrscr();  printf("Utvoreni masuvu\n");  for(i=0;i<10;i++)  {  x[i]=exp(i)+2.5\*i;  printf("x[%d]= %lf\n", i, x[i]);  }  max=x[0];  for(i=0;i<10;i++)  {  if(max<x[i]);  max=x[i];  }  printf("MAX = %lf\n", max);  for(i=0;i<10;i++)  {  y[i]=x[i]-max;  printf("y[%d]= %lf\n", i, y[i]);  }  getch();  return 0;  } |
| **5** | Утворити масив i вивести на екран елементи масиву. *xi* = sin *i / 2ei*. Утворити вектор , *де 1<=i<=12, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[12], y[12];  float min;  clrscr();  printf("Utvoreni masuvu\n");  for(i=0;i<12;i++)  {  x[i]=sin(i)/2\*exp(i);  printf("x[%d]= %lf\n", i, x[i]);  }  min=x[0];  for(i=0;i<12;i++)  {  if(min>[i]);  min=x[i];  }  printf("MIN = %lf\n", min);  for(i=0;i<12;i++)  {  y[i]=x[i]-min;  printf("y[%d]= %lf\n", i, y[i]);  }  getch();  return 0;  } |
| **6** | Утворити масив i вивести на екран елементи масиву. *xi =* sin *i* + 8,54; *уi* = tan *i*. Утворити *zi* = *xi* + *уi*, *де 1<=i<=14, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[14], y[14], z[14];  clrscr();  printf("Utvoreni masuvu\n");  for(i=0;i<14;i++)  {  x[i]=sin(i)+8.54;  printf("x[%d]= %lf\n", i, x[i]);  y[i]=tan(i);  printf("y[%d]= %lf\n", i, y[i]);  }  for(i=0;i<14;i++)  {  z[i]=x[i]+y[i];  printf("z[%d]= %lf\n", i, z[i]);  }  getch();  return 0;  } |
| **7** | Утворити масив i вивести на екран елементи масиву. *xi =* arctan(*i*ln*i)*. Обчислити, *де 1<=i<=10, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[10];  float S, t;  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<10;i++)  {  x[i]=atan(i\*log(i));  printf("x[%d]= %lf\n", i, x[i]);  }  S=0;  for(i=0;i<10;i++)  {  S=S+pow(x[i],2);  }  t=sqrt(S);  printf("Korin iz sumu = %lf", t);  getch();  return 0;  } |
| **8** | Утворити масив i вивести на екран елементи масиву. . Утворити новий вектор, який складається з від’ємних елементів вектора *xi , де 1<=i<=16, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i, v;  float x[16], a[16];  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<16;i++)  {  x[i]=((i+2)/(pow(i,2)+4.57))\*cos(i);  printf("x[%d]= %lf\n", i, x[i]);  }  printf("Masuv z - elementiv x[i]\n");  v=0;  for(i=0;i<16;i++)  if(x[i]<0)  {  a[v]=x[i];  printf("a[%d] = %lf\n", v, a[v]);  v=v+1;  }  getch();  return 0;  } |
| **9** | Утворити масив i вивести на екран елементи масиву. . Знайти найменше значення серед елементів масиву, більших за 0,09.(і=12) | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i, n;  float x[12], a[12];  float min;  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<12;i++)  {  x[i]=fabs(sin(i))/(pow(i,3)+4);  printf("x[%d]= %lf\n", i, x[i]);  }  n=0;  for(i=0;i<12;i++)  if(x[i]>=0.09)  {  a[n]=x[i];  printf("a[%d] = %lf\n", n, a[n]);  n=n+1;  }  min=a[0];  for(i=0;i<n;i++)  if(min>a[i])  min=a[i];  printf("Min = %lf", min);  getch();  return 0;  } |
| **10** | Утворити масив i вивести на екран елементи масиву. *xi = ei*sin*i*. Обчислити , *де 1<=i<=16, крок=1.* | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i;  float x[16];  float z, max, min;  clrscr();  printf("Utvorenuy masuv\n");  for(i=0;i<16;i++)  {  x[i]=exp(i)\*sin(i);  printf("x[%d]= %10.2lf\n", i, x[i]);  }  max=x[0];  for(i=0;i<16;i++)  {  if(max<x[i])  max=x[i];  }  printf("max = %10.2lf\n", max);  min=x[0];  for(i=0;i<16;i++)  {  if(min>x[i])  min=x[i];  }  printf("min = %10.2lf\n", min);  z=max-min;  printf("z = %10.2", z);  getch();  return 0;  } |
| **11** | Дано матрицю. Знайти i вивести на екран індекси максимального елемента. | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i,j;  int m, n;  float x[4][4]={{2.0, 4.0, -6.0, 7.4}, {4.5, -3.0, 5.0, 0.0}, {9.7, 4.2, -3.8, 2.75}, {3.3, -0.6, 0.0, 3.5}};  float max;  clrscr();  printf("Vvedena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  printf("%5.4lf\t", x[i][j]);  }  printf("%s\n"," ");  }  max=x[0][0];  n=0;  m=0;  for(i=0;i<4;i++)  for(j=0;j<4;j++)  if(max<x[i][j])  {  max=x[i][j];  m=i;  n=j;  }  printf("max = %5.2f\n", max);  printf("i= %d\n", m);  printf("j= %d\n", n);  getch();  return 0;  } |
| **12** | Дано матрицю. Вивести індекси нульових елементів.  . | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i,j;  int m[16], n[16];  float x[4][4]={{2.0, 4.0, -6.0, 7.4}, {4.5, -3.0, 5.0, 0.0}, {9.7, 4.2, -3.8, 2.75}, {3.3, -0.6, 0.0, 3.5}};  clrscr();  printf("Vvedena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  printf("%5.4lf\t", x[i][j]);  }  printf("%s\n"," ");  }  printf("Indeksu nulovuh elementiv\n");  for(i=0;i<4;i++)  for(j=0;j<4;j++)  if(x[i][j]==0)  {  m[i]=i;  n[i]=j;  printf("%d\t", m[i]);  printf("%d\n", n[i]);  }  getch();  return 0;  } |
| **13** | Дано матрицю. Обчислити B = *А – АT.* i вивести на екран результат. | #include<stdio.h>  #include<conio.h>  #include<math.h>  #include<iostream>  using namespace std;  int main()  {  int i,j;  float x[4][4]={{2.0, 4.0, -6.0, 7.4}, {4.5, -3.0, 5.0, 0.2}, {9.7, 4.2, -3.8, 2.75}, {3.3, -0.6, 0.3, 3.5}};  float b[4][4];  system ("cls");  printf("Vvedena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  printf("%8.2lf\t", x[i][j]);  }  printf("%s\n"," ");  }  printf("Obchuslena matrutsa\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  b[i][j]=x[j][i];  printf("%5.4lf\t", b[i][j]);  }  printf("%s\n", " ");  }  printf("\n");  for (i=0; i<4;i++)  {  for (j=0;j<4;j++)  b[i][j]=x[i][j]-b[i][j];  }  for (i=0;i<4;i++)  {  for (j=0;j<4;j++)  printf("%5.2lf\t",b[i][j]);  printf("\n");  }  getch();  return 0;  } |
| **14** | Дано матрицю. Обчислити кількість елементів, більших 5-ти і добуток елементів менших від 1 i вивести на екран результати. | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i,j,k;  float x[4][4]={{2.0, 4.0, -6.0, 7.4}, {4.5, -3.0, 5.0, 0.2}, {9.7, 4.2, -3.8, 2.75}, {3.3, -0.6, 0.3, 3.5}};  float d;  clrscr();  printf("Vvedena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  printf("%8.2lf\t", x[i][j]);  }  printf("%s\n"," ");  }  d=1;  for(i=0;i<4;i++)  for(j=0;j<4;j++)  {  if(x[i][j]<1)  d=d\*x[i][j];  }  printf("Dobutok=%5.2f\n", d);  k=0;  for(i=0;i<4;i++)  for(j=0;j<4;j++)  {  if(x[i][j]>5)  k=k+1;  }  printf("Kilkist elementiv >5 dorivnue %d\n", k);  getch();  return 0;  } |
| **15** | Утворити двовимірний масив(4х4) і вивести його на екран у вигляді матриці. . Обчислити загальну суму елементів парних стовпців і вивести результат на екран. | #include<stdio.h>  #include<conio.h>  #include<math.h>  #define pi 3.14  main()  {  int i,j;  float x[4][4];  float S;  clrscr();  printf("Utvorena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  x[i][j]=(i\*tan(pi/5))/2+sin(i\*j);  printf("%5.4lf\t", x[i][j]);  }  printf("%s\n"," ");  }  S=0;  for(i=0;i<4;i++)  for(j=0;j<4;j+=2)  S=S+x[i][j];  printf("Suma S = %5.4lf\n", S);  getch();  return 0;  } |
| **16** | Утворити двовимірний масив(4х4) і вивести його на екран у вигляді матриці. . Кожен елемент другого рядка матриці замінити нулями вивести результат на екран. | #include<stdio.h>  #include<conio.h>  #include<math.h>  #include<iostream>  using namespace std;  int main()  {  int i,j;  float x[4][4], b[4][4];  system ("cls");  printf("Utvorena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  x[i][j]=((pow(i,3.0)+j)\*sin((double)(i\*j)))/2\*i+3.8\*j;  printf("%5.4lf\t", x[i][j]);  }  printf("%s\n"," ");  }  printf("Zamina nulamu\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  if(i==1)  b[i][j]=0;  else  b[i][j]=x[i][j];  printf("%5.4lf\t", b[i][j]);  }  printf("%s\n", " ");  }  getch();  return 0;  } |
| **17** | Утворити двовимірний масив(4х4) і вивести його на екран у вигляді матриці. . Кожен елемент третього стовпця матриці замінити одиницею, вивести результат на екран. | #include<stdio.h>  #include<conio.h>  #include<math.h>  main()  {  int i,j;  float x[4][4], b[4][4];  clrscr();  printf("Utvorena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  x[i][j]=(1+j)\*pow(i,2)+j;  printf("%5.4lf\t", x[i][j]);  }  printf("%s\n"," ");  }  printf("Zaminena nulamu\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  if(j==2)  b[i][j]=1;  else  b[i][j]=x[i][j];  printf("%5.4lf\t", b[i][j]);  }  printf("%s\n", " ");  }  getch();  return 0;  } |
| **18** | Утворити двовимірний масив(4х4) і вивести його на екран у вигляді матриці. . Побудувати вектор з найбільших значень кожного рядка матриці, вивести результат на екран. | #include<stdio.h>  #include<conio.h>  #include<math.h>  int main()  {  int i,j;  float x[4][4];  float c[4];  float max;  printf("Utvorena matrutca\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  x[i][j]=exp((double)(i\*j))\*sin((double)j);  printf("%8.2lf\t", x[i][j]);  }  printf("%s\n"," ");  }  for(i=0;i<4;i++)  {  max=x[i][0];  for(j=0;j<4;j++)  if(max<x[i][j])  {  max=x[i][j];  c[i]=max;  }  printf("c[%d]=%5.2f\n", i, c[i]);  }  getch();  return 0;  } |
| **20 19** | Утворити двовимірний масив (4х4) і вивести його на екран у вигляді матриці. *.* Обчислити кількість додатних елементів у кожному стовпці матриці, вивести результат на екран. | #include<stdio.h>  #include<conio.h>  #include<math.h>  int main ()  {  int i, j, k;  float x[4][4], a[4][4], c[4][4];  printf("Utvorena matrutsa 1\n");  for(i=0;i<4;i++)  {  for(j=0;j<4;j++)  {  x[i][j]=abs(atan((double)(i\*j)));  printf("%8.2lf\t", x[i][j]);  }  printf("%s\n", " ");  }  for(j=0;j<4;j++)  {  k=0;  for(i=0;i<4;i++)  {  if(x[i][j]>=0)  {  k=k+1;  }  }  printf("Kilkist + elementiv u %d stovptsi = %d\n", j, k);  }  getch ();  return 0;  } |