

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 2

з дисципліни “ Основи програмування ”

Тема “Циклічні конструкції та робота з перечисленнями і статичними масивами даних.

|  |  |  |
| --- | --- | --- |
| Виконав  СтудентI курсу  групи КП-51  Медведський Денис Іванович  (*прізвище, ім’я, по батькові*)  варіант №13 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2014

**Мета роботи**

Навчитися працювати зі статичними масивами різних типів даних мови програмування С. Застосувати на практиці різні види циклічних конструкцій для обчислення математичних формул і при роботі з одномірними та багатомірними масивами даних. Застосувати перечислення при вирішенні різнопланових завдань.

**Постановка завдання**

## **Завдання №1. Використання циклічних конструкцій**

За допомогою ітераційних циклічних конструкцій обчислити значення виразу:

## **Завдання №2. Робота з одномірними масивами**

## Реалізувати функції для роботи з одномірними масивами у файлі

Заповнює масив випадковими цілими числами в діапазоні [1..99] (включно)

void fillRand1(int arr[], int size);

Перевіряє чи всі елементи масиву знаходяться у діапазоні [1..99] (включно). Повертає 1, якщо умова задовольняється і 0 - якщо не задовольняється

int checkRand1(int arr[], int size);

Знаходить середнє значення серед елементів масиву

float meanValue(int arr[], int size);

Знаходить максимальний серед елементів масиву

int maxValue(int arr[], int size);

Знаходить індекс першого елементу, значення якого найближче до середнього значення серед елементів масиву

int meanIndex(int arr[], int size);

Знаходить індекс першого максимального елемента масиву

int maxIndex(int arr[], int size);

Знаходить значення, яке найчастіше зустрічається у масиві. Якщо таких декілька, повертає більше з них

int maxOccurance(int arr[], int size);

Розміри arr1, arr2 і res одинакові. Перевіряється різниця між відповідними елементами масивів arr1 і arr2. Ця різниця записується у відповідний елемент масиву res. Якщо всі різниці рівні 0 (тобто масиви arr1 і arr2 одинакові, то функція повертає 1, інакше - 0).

int diff(int arr1[], int arr2[], int res[], int size);

Розміри arr1, arr2 і res одинакові. Виконати відповідну арифметичну операцію над елементами масивів arr1 і arr2 і записати у відповідний елемент масиву res результат ділення

void div(int arr1[], int arr2[], int res[], int size);

Розміри arr1, arr2 і res одинакові. Виконати відповідну операцію відношення над елементами масиву lteq - less than or equals

int lteq(int arr1[], int arr2[], int size);

Розміри arr1, arr2 і res одинакові. Виконати відповідну логічну операцію над елементами масивів arr1 і arr2 і записати результат у масив res. Всі елементи масиву знаходяться у діапазоні [0..1] (включно) lor - logical OR

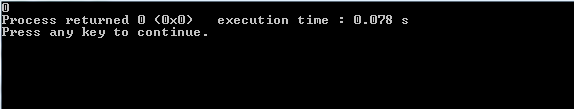
void lor(int arr1[], int arr2[], int res[], int size);

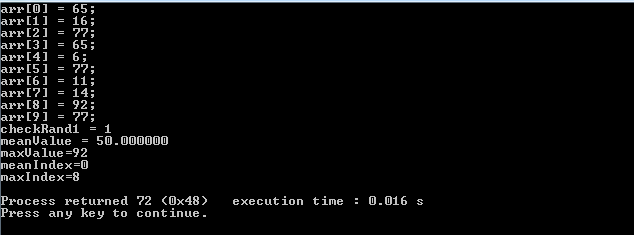
## Завдання №3. Моделювання кінцевого автомата

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **\** | **5** | **13** | **23** | **203** |
| **0** | 5,0 | pop,1 |  | 551,2 |
| **1** |  | pop,2 | 552,1 | **repeat**, 2 |
| **2** | continue,3 | 555,3 |  | 550,1 |
| **3** |  | 553,0 | break | break |

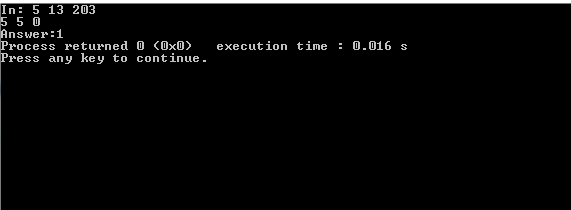
Прикладт роботи програм:

1)



2) 

3)



**Тексти коду програм**

**Завдання 1.**

|  |
| --- |
| loops.c |
| #include <stdio.h>  #include <math.h>  double calc(int n, int m)  {  int i,j,a=0;  int c=0;  int b=0;  for (i=1; i<=n; i++)  {  a=1+i;  for(j=1; j<=m; j++)  {  b=cos((j\*(M\_PI))/2)+b;  }  c=(a\*b)+c;  }  printf("%i", c);  return c;  } |

**Завдання 2.**

|  |
| --- |
| array.c |
| **#include <stdio.h>**  **#include <stdlib.h>**  **#include <math.h>**  **#include <time.h>**  **void fillRand1(int arr[], int size)**  **{**  **int i;**  **time\_t t;**  **srand((unsigned) time(&t));**  **for (i = 0; i<size; i++)**  **{**  **arr[i] = 1 + rand() % 99;**  **printf("arr[%i] = %i;\n", i, arr[i]);**  **}**  **}**  **int checkRand1(int arr[], int size)**  **{**  **int answer = 1;**  **int i;**  **for (i = 0; i<size; i++)**  **{**  **if ((arr[i]<1) || (arr[i]>99)){**  **answer = 0;**  **break;**  **}**  **}**  **return answer;**  **}**  **float meanValue(int arr[], int size)**  **{**  **float average=0;**  **int i;**  **for (i=0; i<size; i++)**  **{**  **average+= arr[i];**  **}**  **average/=size;**  **return average;**  **}**  **int maxValue(int arr[], int size)**  **{**  **int max=0,i;**  **for (i=1; i<size; i++)**  **{**  **if (arr[i]>arr[max])**  **max=i;**  **}**  **return(arr[max]);**  **}**  **int meanIndex(int arr[], int size)**  **{**  **int i,min=0;**  **int answer=0;**  **float mv = (meanValue(arr,size));**  **min = 0;**  **for(i=1; i<size; i++)**  **{**  **if(abs(mv-arr[i])<abs(mv-arr[min]))**  **{**  **min=i;**  **}**  **}**  **return min;**  **}**  **int maxIndex (int arr[], int size)**  **{**  **int max = maxValue(arr, size);**  **int stop = 0;**  **int answer;**  **int i=0;**  **while (stop==0)**  **{**  **if (max ==arr[i])**  **{**  **stop = 1;**  **answer = i;**  **}**  **i++;**  **}**  **}**  **int maxOccurance(int arr[], int size)**  **{**  **int i;**  **int num[size];**  **for (i=0; i<size; i++)**  **{**  **num[i]=0;**  **}**  **int j;**  **for (i=0; i< size; i++)**  **{**  **for (j=0; j< size; j++)**  **{**  **if (arr[i] == arr[j])**  **num[i]++;**  **}**  **}**  **int top = num[0];**  **int answer = arr[0];**  **for (i=0; i< size; i++)**  **{**  **if ((top < num[i]) || ((top == num[i]) && (answer < arr[i])))**  **{**  **top = num[i];**  **answer = arr[i];**  **}**  **}**  **return answer;**  **}**  **int diff(int arr1[], int arr2[], int res[],int meanIndex, int arr[], int size)**  **{**  **int value = 0;**  **int i;**  **for (i=1; i<size; i++)**  **{**  **if (arr1[i]-arr2[i]==0)**  **{**  **res[i]=0;**  **value++;**  **}**  **else**  **{**  **res[i]=arr1[i]-arr2[i];**  **}**  **}**  **if (value=size)**  **{return 0;**  **}**  **return 0;**  **}**  **void divv(int arr1[], int arr2[], int res[], int size)**  **{**  **int i;**  **for (i = 0; i<size; i++)**  **{**  **res[i] = arr1[i]/arr2[i];**  **}**  **}**  **int lteq(int arr1[], int arr2[], int size)**  **{**  **int i;**  **for (i = 0; i<size; i++)**  **{**  **if (arr1[i] <= arr2[i])**  **return 0;**  **}**  **return 1;**  **}**  **void lor(int arr1[], int arr2[], int res[], int size)**  **{**  **int i;**  **for (i=0; i<size; i++)**  **{**  **res[i] = arr1[i] | arr2[i];**  **}**  **}** |

**Завдання 3.**

|  |
| --- |
| automata.c |
| #include <stdio.h>  #include <stdlib.h>  #include <math.h>  enum commands {Pop=-10, Push, Continue, Repeat, Break};  struct move\_info  {  int command;  int what\_to\_push;  int state;  };  int run(int moves[], int movesLen, int res[], int resLen)  {  struct move\_info what\_to\_do[4][4];  int i,j;  what\_to\_do[0][0].command = Push;  what\_to\_do[0][0].what\_to\_push=5;  what\_to\_do[0][0].state = 0;  what\_to\_do[0][1].command = Pop;  what\_to\_do[0][1].state = 1;  what\_to\_do[0][2].command = Break;  what\_to\_do[0][3].command = Push;  what\_to\_do[0][3].what\_to\_push=551;  what\_to\_do[0][3].state = 2;  what\_to\_do[1][0].command = Break;  what\_to\_do[1][1].command = Pop;  what\_to\_do[1][1].state = 1;  what\_to\_do[1][2].command = Push;  what\_to\_do[1][2].what\_to\_push=552;  what\_to\_do[1][2].state = 1;  what\_to\_do[1][3].command = Repeat;  what\_to\_do[1][3].state = 1;  what\_to\_do[2][0].command = Continue;  what\_to\_do[2][0].state = 3;  what\_to\_do[2][1].command = Push;  what\_to\_do[2][1].what\_to\_push=555;  what\_to\_do[2][1].state = 3;  what\_to\_do[2][2].command = Break;  what\_to\_do[2][2].command = Push;  what\_to\_do[2][2].what\_to\_push = 550;  what\_to\_do[2][2].state = 1;  what\_to\_do[3][0].command = Break;  what\_to\_do[3][2].command = Break;  what\_to\_do[3][1].command = Push;  what\_to\_do[3][1].what\_to\_push = 553;  what\_to\_do[3][1].state = 0;  what\_to\_do[3][3].command = Break;  for (i=0; i<resLen; i++)  {  res[i]=0;  }  // just put some nums  //  int state\_now=0;  int pos\_now=0;  int finish=0; // bool  for (i=0; i<movesLen; i++)  {  if (returner(moves[i])!=NAN)  {  switch(what\_to\_do[state\_now][returner(moves[i])].command)  {  case Push:  {  res[pos\_now] = what\_to\_do[state\_now][returner(moves[i])].what\_to\_push;  state\_now = what\_to\_do[state\_now][returner(moves[i])].state;  pos\_now++;  break;  }  case Pop:  {  if (pos\_now!=0)  {  res[pos\_now]=0;  pos\_now--;  }  else  finish = 1; // true  break;  }  case Continue:  {  state\_now = what\_to\_do[state\_now][returner(moves[i])].state;  break;  }  case Repeat:  {  state\_now=what\_to\_do[state\_now][returner(moves[i])].state;  i--;  break;  }  case Break:  {  finish = 1;  break;  }  }  }  else  finish = 1;  //  if (finish==1)  {  break;  }  }  i=resLen-1;  int answer=0;  while((res[i]==0) && (i>=0))  {  answer++;  i--;  }  for (i=0;i<resLen; i++)  {  printf("%i ", res[i]);  }  return answer;  }  int returner(int x)  {  if (x==10)  return 0;  else if (x==14)  return 1;  else if (x==19)  return 2;  else if (x==109)  return 3;  else  return NAN;  }  int main()  {  int res[2];  int i;  int rl =3;  int x[2];  x[0]=5;  x[1]=13;  x[2]=203;  printf("In: ");  for (i=0;i<3; i++)  {  printf("%i ", x[i]);  }  printf("\n");  int xl=2;  printf("\nAnswer:%i", run(x,xl,res,rl));  return 0;  } |