Центральноукраїнський національний технічний унiверситет

Кафедра програмування комп’ютерних систем і мереж

Дисципліна: Базові методології та технології програмування

Звiт

з лабораторної роботи №6

Тема: Програмна обробка алгоритмічних конструкцій циклічної структури

Мета роботи: Навчитись складати алгоритми розв’язку задач, які вміщують циклічний обчислювальний процес, та програми для їх реалізації; отримати навики у визначенні та використанні оператора цикла; ознайомитися з ітераційними процесами.

Виконав: ст. гр. КМ-20

Калиновський В.М.

Перевірила: асистент кафедри ПКСМ

Тесленко О.Є.

Варіант 3

Дата виконання: 30.10.2020

Дата оформлення: 03.11.2020

**Кропивницький 2020 р.**

**Завдання:** 1)Формалізувати зміст задачі, скласти блок-схему алгоритму та програму обома мовами для обчислення на ЕОМ таблиці значень функції вигляду F(x) на інтервалі [a, b] з кроком , де m – задане ціле число; там, де потрібно, врахувати область визначення функції; розробити два варіанти програми: перший – значення змінної в точка розраховується за формулою , скористатись циклом з параметром та вивести дані у формі «*Для x[1]=<значення>: функція y[1]=<значення>*»; другий – значення змінної в точка розраховується за формулою , скористатись циклом з передумовою та вивести у форматі таблиці, де X та Y записані у заголовку таблиці, а отриманні значення роздрукувати нижче, відповідно, під ними; за третім варіантом дана функція , інтервал від 1,2 до 2 та кількість точок – 10. 2)Скласти блок-схему алгоритму та програму, яка обчислює функцію, розкладену в ряд Маклорена з заданою точністю; на друк вивести: функцію, її значення при розкладанні функції в ряд Маклорена, кількість елементів, які врахувались при розрахунках для досягнення заданої точності; зробити перевірку рішення; за третім варіантом дана функція та точність .

**Робота.** Після ознайомлення із завданням я складаю алгоритм для програми першого завдання:

1. Ввести значення початку (a), кінця (b) відрізка та кількість точок (m) на відрізці
2. Розрахувати шаг h за формулою
3. Розрахувати значення X та Y залежно від шагу
4. Цикл повторюється, поки не буде проведено кількість розрахунків, що залежить від кількості точок
5. Вивести всі ці значення

Код першого варіанта оформлення на С написано у таблиці 1, другий у таблиці 2, перший на Паскалі у таблиці 3 та другий у таблиці 4. Результати виконання всіх програм зображено на рисунку 1.

Алгоритмом другої програми є:

1. Введення значення межі розрахунку () та значення функції F(x)
2. Введення значень , n,
3. Розраховую за формулою
4. Цикл повторюється, поки
5. Після виводиться результат розрахунку

Код другої програми, написаної на С виведено у таблиці 5, а на мові Паскаль у таблиці 6, результати виконання програм виведено на рисунку 2.

Таблиця 1. Код для виконання програми першого варіанту виведення на С

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <Windows.h>  void f(float x);  float y;  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  float a = 1.2; //Начало отрезка  float b = 2; //Конец отрезка  float m = 10; //Количество точек на отрезке  float xi = a; //Выводимое число  float h = (b - a) / m; //Определение шага отрезка    float xy; //Нужна для вывода того же X значения, но для переменной Y  int i = 1;  for (i; i <= 10; i++) {  xi = xi + h;  xy = xi;  f(xi);  printf("Для X[%2i] = %4.2f: функция Y[%4.2f] = %4.3f\n", i, xi, xy, y);  }  system("pause");  return 0;  }  void f(float x) {  /\*Так как с синусом программа отказывается раотать: при указании значения 1/2 вообще  его не считает, а при указании 0.5 выдаёт совсем странные результаты, я сам расчитал  значение sin(1/2) и подставил это готовое значение в формулу\*/  y = x - 2 + 0.008;  } |

Таблиця 2. Код для виконання програми другого варіанту виведення на С

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <Windows.h>  void f(float x);  float y;  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  float a = 1.2; //Начало отрезка  float b = 2; //Конец отрезка  float m = 10; //Количество точек на отрезке  float x = a; //Выводимое число  float h = (b - a) / m; //Определение шага отрезка  printf("|\_\_\_\_X\_\_\_\_|\_\_\_\_\_Y\_\_\_\_|\n");  int i = 1;  while (i <= 10) {  x = x + h;  f(x);  if (y < 0) {  printf("|\_\_%5.3f\_\_|\_\_%5.3f\_\_|\n", x, y);  }  else {  printf("|\_\_%5.3f\_\_|\_\_\_%5.3f\_\_|\n", x, y);  }  i++;  }  system("pause");  return 0;  }  void f(float x) {  /\*Так как с синусом программа отказывается раотать: при указании значения 1/2 вообще  его не считает, а при указании 0.5 выдаёт совсем странные результаты, я сам расчитал  значение sin(1/2) и подставил это готовое значение в формулу\*/  y = x - 2 + 0.008;  } |

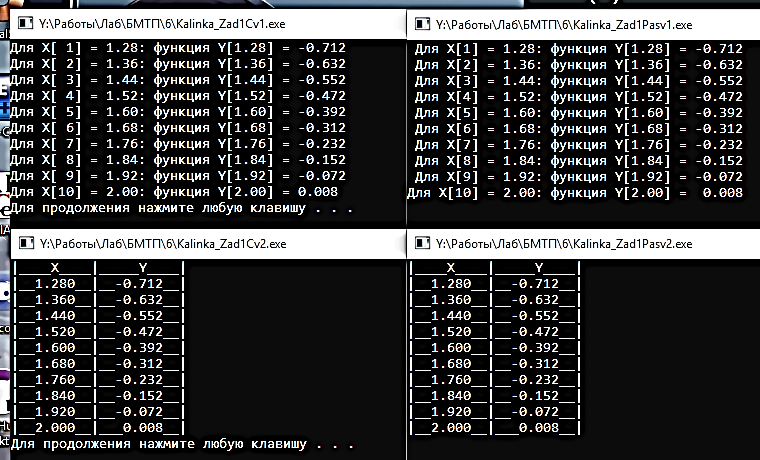
Таблиця 3. Код для виконання програми першого варіанту виведення на Паскалі

|  |
| --- |
| **program** Kalinka\_Zad1Pasv1;  **uses** crt;  **var** a, b, m, xi, h, xy, y:real;  **function** f(x:real):real;  **begin**  y:=x-2+0.008;  **end**;  **begin**  a:=1.2;  b:=2;  m:=10;  xi:=a;  h:=(b-a)/m;  **for** i:integer:=1 **to** 10 **do**  **begin**  xi:=xi+h;  xy:=xi;  f(xi);  **if** (i<10) **then** writeln(' Для X[', i, '] = ', xi:4:2, ': функция Y[', xy:4:2, '] = ', y:5:3) **else** writeln('Для X[', i, '] = ', xi:4:2, ': функция Y[', xy:4:2, '] = ', y:5:3);  **end**;  readkey;  **end**. |

Таблиця 4. Код для виконання програми другого варіанту виведення на Паскалі

|  |
| --- |
| **program** Kalinka\_Zad1Pasv2;  **uses** crt;  **var** a, b, m, x, h, y:real; i:integer;  **function** f(x:real):real;  **begin**  y:=x-2+0.008;  **end**;  **begin**  a:=1.2;  b:=2;  m:=10;  x:=a;  h:=(b-a)/m;  writeln('|\_\_\_\_X\_\_\_\_|\_\_\_\_\_Y\_\_\_\_|');  i:=1;  **while** i<=10 **do**  **begin**  x:=x+h;  f(x);  **if** (y<0) **then** writeln('|\_\_', x:5:3, '\_\_|\_\_', y:5:3, '\_\_|') **else** writeln('|\_\_', x:5:3, '\_\_|\_\_\_', y:5:3, '\_\_|');  i:=i+1;  **end**;  readkey;  **end**. |

Рисунок 1. Результат виконання всіх програм першого завдання



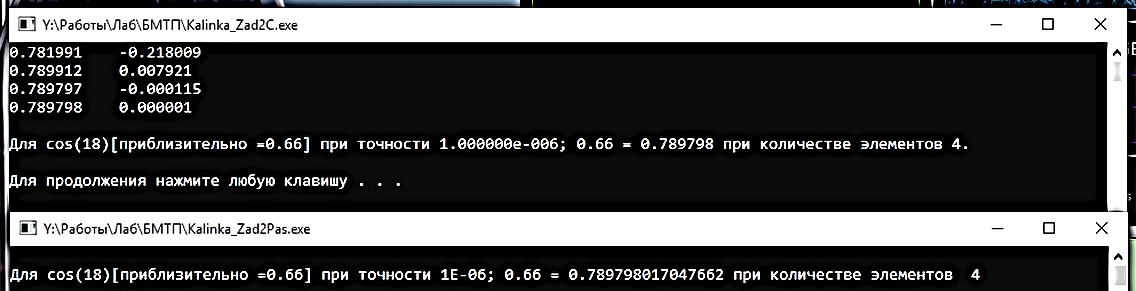
Таблиця 5. Код для виконання програми другого завдання на С

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <Windows.h>  const float eps = 1E-6;  const float cosiks = cos(18);  int factorial(int n) { //Объявляю функцию, которая будет считать факториал  if (n == 1) {  return 1;  }  else {  return n \* factorial(n - 1);  }  }  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  double Maklorcosiks = 1; //Здесь "Maklorcosiks" представляет собой косинус икса в ряде Маклорена  int n = 0; //Количество элементов  double an;    do {  n++; //Увеличиваю номер на единицу  int n2 = 2 \* n; //Вспомогательная переменная    double oneinn = pow(-1, n); //Первое число - возвожу -1 в степень н  double x2n = pow(cosiks, n2); //Второе число - возвожу икс в степень 2н  int fact2n = factorial(n2); //Третье число - нахожу факториал числа 2н  an = oneinn \* (x2n / fact2n); //Подставил все три числа в формулу для нахождения a\_n  Maklorcosiks = Maklorcosiks + an; //Увеличиваю число "an" в "b" раз  printf("%lf %lf\n", Maklorcosiks, an); //Для проверка исправности программы  } while (fabs(an) >= eps);  /\*Если нужно выполнять цикл до тех пор, пока он не станет меньше чего-то,  значит он выполняется, пока он больше этого чего-то и стремится к нему (уменьшается),  поэтому цикл выполняется пока an больше eps, и с каждым разом an становиться всё  меньше, и когда an станет меньше eps - цикл закончится.\*/  printf("\nДля cos(18)[приблизительно =%.2f] при точности %e; %.2f = %lf при количестве элементов %i.\n\n", cosiks, eps, cosiks, Maklorcosiks, n);  system("pause");  return 0;  } |

Таблиця 6. Код для виконання програми другого завдання на Паскалі

|  |
| --- |
| **program** Kalinka\_Zad1Pasv1;  **uses** crt;  **var** eps, cosiks, Maklorcosiks, an, oneinn, x2n, fact2n, n, n2, nprov, nprov1, nprov2, nprov3, nprov4:real;  **function** factorial(n:real):real;  **begin**  **if** (n=1) **then** factorial:=1 **else** factorial:=n\*factorial(n-1);  **end**;  **begin**  eps:=1E-6;  cosiks:=cos(18);  Maklorcosiks:=1;  n:=0;  **repeat**  n:=n+1;  n2:=2\*n;    nprov:=n; nprov:=nprov/1; nprov1:=4; nprov2:=4; nprov3:=4; nprov4:=4;    **if** nprov/1=1 **then** nprov1:=5;  **if** nprov/2=1 **then** nprov2:=5;  **if** nprov/3=1 **then** nprov3:=5;  **if** nprov/4=1 **then** nprov4:=5;    **if** nprov1=5 **then** oneinn:=-1;  **if** nprov2=5 **then** oneinn:=1;  **if** nprov3=5 **then** oneinn:=-1;  **if** nprov4=5 **then** oneinn:=1;    x2n:=exp(n2\*ln(cosiks));  fact2n:=factorial(n2);  an:=oneinn\*(x2n/fact2n);  Maklorcosiks:=Maklorcosiks+an;  **until** abs(an)<=eps;  writeln;writeln('Для cos(18)[приблизительно =', cosiks:4:2, '] при точности ', eps ,'; ', cosiks:4:2, ' = ', Maklorcosiks, ' при количестве элементов ', n:2:0);  readkey;  **end**. |

Рисунок 2. Результат виконання всіх програм другого завдання



Перевірю роботу програми у калькуляторі: . Мій ряд , тому , а . Оскільки . Повний розрахунок, порахований вручну, буде прикладено до звіту. Як ми бачимо, числа не зовсім зівпали, але такій похибці ми можемо дати причину погано сформульованого самого ряда. Але і мій результат і результат програми ідентичні. Можна вважати програму робочою.

Слід відмітити, чому погано так написана програма на Паскалі. Річ у тім, що на Паскалі не можна возводити ні до ніякої степені , тому приходиться присвоювати мінус значенню залежно від самого номера n. Знаючи, скільки елементів у ряді з програми на С, я зробив особисту перевірку для кожного номера. Тільки так можна заставити програму правильно порахувати значення.

Перевірку першого завдання можна й не проводити, оскільки всі розрахунки можна проводити у голові, але розв’язок я все-таки у додатку написав – і він співпав з результатами програми. Буду вважати теж робочою.

Відповіді на контрольні питання:

1. Дані якого типу можна вживати як параметр циклу?

integer

1. Указати призначення та правила організації циклів у структурному програмуванні.

Цикли призначені для багаторазового використання однієї частини коду методом повторення. Повторення відбувається до тих пір, доки умова задовільняє значенню. Це значення і є параметром. Правилом організації циклу є обов’язкове зазначення параметрів у циклі. У разі їх невикористання цикл може бути або нескінченним або взагалі не працювати, в залежності від типа циклу

1. Як працює цикл while?

Він перевіряє умову й виконує цикл до тих пір, доки умова задовільняє виконання циклу

1. Як працює цикл repeat?

Він без перевірки починає виконання циклу й після першого виконання перевіряє умову й продовжується доти, доки параметр задовільняє умову. На відмінну від попереднього циклу, цей цикл виконується хоча б раз.

1. Вказати послідовність дій, які виконуються при організації циклічних розрахунків з указаною кількістю повторів?

Ініціалізувати параметр, дати йому початкове значення, зазначити межу та зазначити шаг зміни лічильника

1. Які обмеження накладаються на оператор циклу for?

У ньому можна зазначити тільки можливу конкретну кількість повторів на відміну від попередніх двох типів циклів, які виконуються стільки разів, доки вони задовільняють умові

1. Які оператори циклів використовуються для виконання повторювальних операцій в мові С?

while; do – while; for

1. Чим відрізняються цикли з передумовою та послідуючою умовою в мовах С та Паскаль?

C: передумова while () {}

послідуюча умова do {} while;

Pascal: передумова while <> begin .. end;

послідуюча умова repeat begin .. end; until <> ;

1. Який формат запису оператора for у мові C?

for (лічильник; межа\_лічильника; шаг\_зміни\_лічильника) {}

1. Що таке вкладені цикли?

Цикли, які викликаються та виконуються у циклі

1. Як можна задати вічний цикл? Навести приклади

Задати цикл for без передачі параметрів, або цикл while з передписаним параметром, який всередині циклу не змінюється, тому не може завершитися:

* 1. for (;;) {}
  2. i = 1; do {printf(«Hello? World!»);} while (i <= 2);

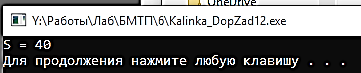
1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

Код програми написано у таблиці 7, а результат на рисунку 3

Таблиця 7. Код программи додаткового завдання під номером 12

|  |
| --- |
| #include <stdio.h>  void main() {  int S = 0;  int k;  int i;  for (i = 1; i <= 5; i++) {  k = 1 + i;  S = S + 2 \* k;  }  printf("S = %i\n", S);  system("pause");  } |

Рисунок 3. Результат виконання додаткового завдання під номером 12



1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

Код першого циклу написано у таблиці 8, другого у 9, третього у 10, четвертого у 11, їх результати на рисунку 4.

Таблиця 8. Код на представлений перший цикл

|  |
| --- |
| int main() {  float x;    printf("X = ");  scanf("%f", &x);  while (x < 1.3) {  x = x \* x;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

Таблиця 9. Код на представлений другий цикл

|  |
| --- |
| #include <stdio.h>  #include <math.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (fabs(x) >= 1) {  x--;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

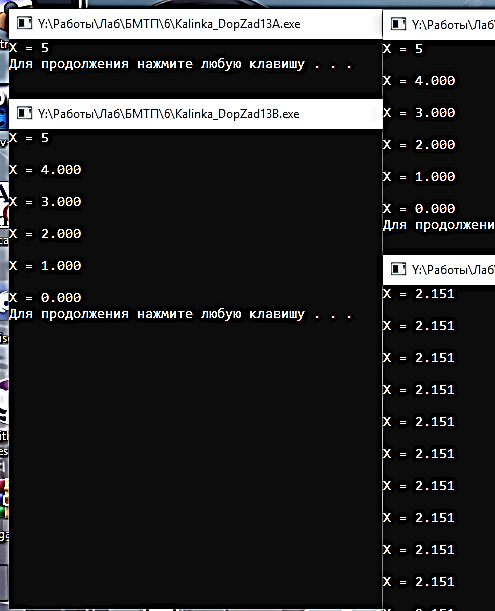
Таблиця 10. Код на представлений третій цикл

|  |
| --- |
| #include <stdio.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (2 \* x > x) {  x--;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

Таблиця 11. Код на представлений четвертий цикл

|  |
| --- |
| #include <stdio.h>  #include <math.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (x \* x >= 0) {  x = sin(x)+1.315;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

Рисунок 4. Результат виконання програм із 13 завдання



Для кожного циклу по різному. Наприклад, для четвертого циклу незалежно від значення X він буде нескінченно зациклюватись; для першого циклу, якщо взяти значення більше 1.3 – він не починається, при значенні менше 1 – він буде нескінченним (як і четвертий), а при значенні між 1 та 1.3 – він майже відразу закінчується; другий та третій цикли працюють адекватно й рахують від заданого значення поступово до 1.

1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

Для розв’язання цієї задачі я написав три варіанти програми, які описані у таблицях 12, 13, 14, а результат на рисунку 5.

Таблиця 12. Перший варіант програми 14 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (fabs(x) + 1 > 0.793) {  x++;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

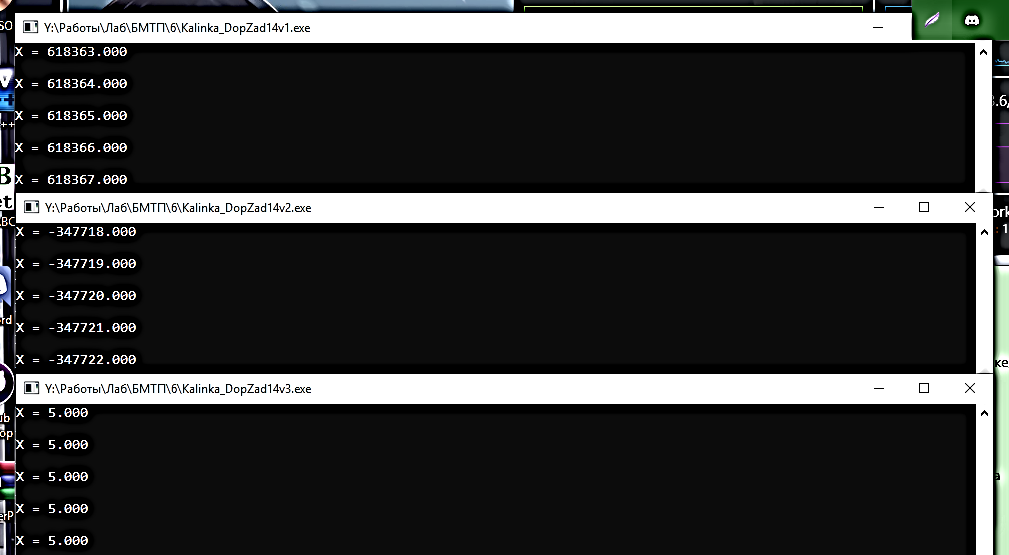
Таблиця 13. Другий варіант програми 14 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (fabs(x) + 1 > 0.793) {  x--;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

Таблиця 14. Третій варіант програми 14 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  int main() {  float x;  printf("X = ");  scanf("%f", &x);  while (fabs(x) + 1 > 0.793) {  x--;  x++;  printf("\nX = %5.3f\n", x);  }  system("pause");  return 0;  } |

Рисунок 5. Результати виконання всіх варіантів програм 14 завдання



Як ми бачимо – ні. Так як у завданні не було зазначено, від чого залежить зміна параметра циклу, прийшлося писати три програми на всі три можливі випадки. Перший випадок – зміна X на 1 одиницю більше: цикл зациклюється з постійним збільшенням на одиницю; другий випадок - зміна X на 1 одиницю менше: цикл зациклюється з постійним зменшенням на одиницю; третій випадок - зміна X на 1 одиницю менше, а потім її збільшення: цикл зациклюється, виводячи тільки значення 5.000. Це відбувається, оскільки незалежно від заданого значення до нього додається одиниця – це число буде ЗАВЖДИ БІЛЬШЕ 1, а параметр порівнюється з числом, яке менше одиниці, тому цикл буде нескінченним.

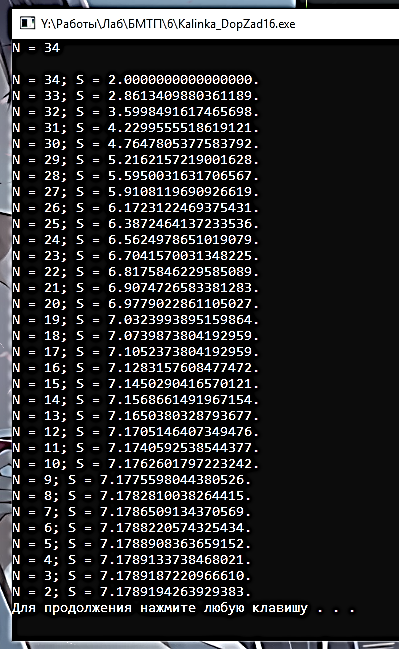
1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

Код для розв’язку завдання написано у таблиці 15, а результат на рисунку 6

Таблиця 15. Код програми 16 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  double n;  int main() {  double S = 1;  printf("N = ");  scanf("%lf", &n);  double an = 1;  printf("\n");  for (n; n != 1; n--) {  S = S + an;  double b = pow(n - 1, 5) / pow(n, 5);  an = an \* b;  printf("N = %.0lf; S = %.16lf.\n", n, S);  }  system("pause");  return 0;  } |

Рисунок 6. Результат виконання програми завдання 16



**ПОЯСНЕННЯ!** *Кожна строка виводиться тільки для перевірки працездатності програми.*

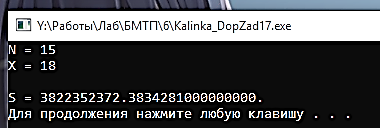
1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

Код для розв’язку завдання написано у таблиці 16, а результат на рисунку 7

Таблиця 16. Код програми 17 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  double n;  double x;  int factorial(int n) {  if (n == 1) {  return 1;  }  else {  return n \* factorial(n - 1);  }  }  int main() {  double S = x;  printf("N = ");  scanf("%lf", &n);  printf("X = ");  scanf("%lf", &x);  double an = x;  printf("\n");  for (n; n != 1; n--) {  S = S + an;  double xn = pow(x, n);  int n1 = n - 1;  int fact\_n1 = factorial(n1);  int fact\_n = factorial(n);  double x\_n1 = pow(x, n1);  double b = (xn \* fact\_n1) / (fact\_n \* x\_n1);  an = an \* b;  }  S = S + an + x;  printf("S = %.16lf.\n", S);  system("pause");  return 0;  } |

Рисунок 7. Результат виконання програми завдання 17



**ПРИМІТКА!** *При використанні цієї програми краще не користуватись великими числами!*

1. Написати код (завдання написано у самій лабораторній роботі) й дати відповідь.

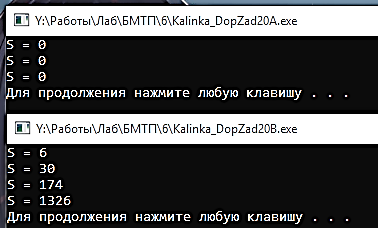
У завданні дано 2 фрагменти. Перший я відтворив у програмі таблиці 17, другий у таблиці 18, а їх результати на рисунку 8

Таблиця 17. Код першого варіанта програми 20 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <Windows.h>  void main() {  int S = 0;  int n = 1;  int i;  for (i = 1; i <= 3; i++) {  int s = S + i;  n = n \* s;  printf("S = %i\n", S);  }  system("pause");  } |

Таблиця 18. Код другого варіанта програми 20 завдання

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <Windows.h>  void main() {  int P = 1;  int s = 2;  int i;  for (i = 1; i <= 4; i++) {  P = P \* 2 \* i;  int S = S + 3 \* P;  printf("S = %i\n", S);  }  system("pause");  } |

Рисунок 8. Результат виконання програм завдання 20

**Висновок:** Ми навчились складати алгоритми розв’язку задач, які вміщують циклічний обчислювальний процес, та програми для їх реалізації; отримали навики у визначенні та використанні оператора цикла; ознайомилися з ітераційними процесами.

*Відразу відмічу, що всі математичні рішення й блок-схеми до всіх задач написані від руки й будуть прикладені до звіту.*