Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Лабораторна робота №12

Виконав:
Студент групи АП-11
Білий Анатолій Іванович

Прийняв:

Чайковський І.Б.

Львів – 2024

«Оператори циклу»

Мета роботи: ознайомитися з особливостями функціонування операторів циклу та навчитись їх використовувати у процесі програмування.

Теоретичні відомості

У мові С, як і в інших мовах програмування, оператори циклу служать для багаторазового виконання послідовності операторів до тих пір, поки виконується деяка умова. Умова може бути встановлена заздалегідь (як в операторі for) або змінюватися при виконанні тіла циклу (як в while або do-while).

Цикл for. У всіх процедурних мовах програмування цикли for дуже схожі. Однак в С цей цикл особливо гнучкий і потужний. Загальна форма оператора for наступна:

for (необов'язковий вираз 1;необов'язковий вираз 2;необов'язковий вираз 3)

оператор;

Вираз 1 призначений для ініціалізації циклу і виконується один раз. Здебільшого тут задаються початкові значення змінних циклу.

Другий вираз задає виконувану перед кожною ітерацією перевірку, за якою здійснюється вихід із циклу. Третій вираз задає приріст, виконуваний після кожної ітерації. Вирази 1 і 3 або один із них можуть бути опущені. У цьому випадку опускати символ **;** не можна. Відсутність першого виразу робить цикл незкінченним, що вимагає передбачення шляхів виходу з нього іншим способом. Якщо за допомогою одного одного із виразів необхідно виконати декілька дій, то використовують запис операторів розділених комою.

У наступному прикладі в циклі for виводяться на екран числа від 1 до 100:

```
#include <stdio.h>
int main (void)
{
 int x;
for (x = 1; x <= 100; x ++)
printf ( "% d", x);
return 0;
}
```

У цьому прикладі параметр циклу x ініціалізований числом 1, а потім при

кожній ітерації порівнюється з числом 100. Поки змінна x менше 100, викликається функція printf () і цикл повторюється. При цьому x збільшується на 1 і знову перевіряється умова циклу x <= 100. Процес повторюється, поки змінна x не стане більше 100. Після цього процес виходить з циклу, а управління передається оператору, наступному за ним. У цьому прикладі параметром циклу є змінна x, при кожній ітерації вона змінюється і перевіряється в секції умови циклу.

Цикл while. Синтаксис данного циклу:

while (вираз)

оператор;

Виконання оператора повторюється, доки значення виразу залишається ненульовим. Перевірка виконується перед кожним виконанням оператора while. Вираз може бути арифметичного або логічного типів. Цикл може бути виконаний один раз, декілька разів або не виконуватися жодного разу.

Для прикладу наведено програму для виводу на екран степеня двійки до 1024.

```
#include<stdio.h>
main()
{
int i=2;
while (i<=1024)
  {
        i = i*2;
        printf("%d\n",i);
   }
}
```

Наступна програма демонструє виведення на екран значення числа k =50.

```
#include<stdio.h>
main()
{
int j=0,k=0;
while(j<5)
 {
        k+=10;
        j++;
 }
printf("k=%d\n",k);
}
```

Цикл do while. В циклі **do while** перевірка умови здійснюється після

виконання тіла циклу. Синтаксис циклу:

do *оператор* ***while*** *(вираз);*

Виконання оператора повторюється доти, поки значення виразу залишається ненульовим. Перевірка виконується після кожного виконання оператора do. Вираз може бути арифметичним або логічним. Оператор у циклі **do while** на відміну від циклу **while** буде виконуватися хоча б один раз завжди. Наступна програма демонструє виведення на екран значення чисел b,c,a.

```
#include<stdio.h>
main()
{
int a=2, b=10,c;
        do
        {
        b=b+a; c=10*a; a++;
        }
        while(a<5);
printf("\nb=%d c=%d a=%d",b,c,a);
}
```

Нижченаведена програма демонструє процес введення числа до того моменту, поки його значення не буде співпадати з магічним числом.

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
int magic;
int guess;
magic=rand();
printf("vgaday chuslo:");
scanf("%d",&guess);
if(guess==magic){
printf("**virno**");
printf("magichne chuslo rivne %d",magic);
   }
else {
printf("nevirno\n");
while (guess!=magic){
printf("vgaday chuslo:");
scanf("%d",&guess);
printf("nevirno\n");
      }
   }
}
```

Наступна програма після введення кожного числа здійснює підрахунок кількості спроб та виводить це значення..

```
#include<stdio.h>
#include<stdlib.h>

int main(void)
{
int magic;
int  guess;
int m=1;
magic=rand();

printf("vgaday chuslo:");
scanf("%d",&guess);
if(guess==magic){
printf("**virno**");
printf("magichne chuslo rivne %d",magic);
   }

else {
printf("nevirno\n");
printf("m=%d\n",m);
while (guess!=magic){
          m++;
printf("vgaday chuslo:");
scanf("%d",&guess);
printf("nevirno\n");
printf("m=%d\n",m);
      }
   }
}
```

У наступній програмі введено обмеження на максимальну кількість спроб введення числа, яке дорівнює значенню рандомно згенерованого магічного числа. Після використання к-сті заданих спроб, виводиться значення магічного числа.

```
#include  <stdio.h>
#include  <stdlib.h>
int main(void)
{
int magic;
int guess;
int m=1; /*кількість спроб*/
magic=rand();
```

```
printf("vgaday chuslo:");
scanf("%d",&guess);
if(guess==magic){
printf("**virno**");
printf("magichne chuslo rivne %d", magic);
   }

else {
printf("nevirno\n");
printf("m=%d\n",m);
while (guess!=magic){
m++;
printf("vgaday chuslo:");
scanf("%d", &guess);
printf("nevirno\n");
printf("m=%d\n",m);
if(m>9){
printf("magic=%d\n", magic);
break;
      }
      }
   }
}
```

Хід роботи:

1. Ознайомитися з теоретичними відомостями.

2. Здійснити виконання усіх прикладів, представлених у теоретичних відомостях, після чого представити скріни їх коду та результати їх виконання у звіті.

3. Виконати нижченаведену програму для обчислення таблиці переведення температури за шкалою Фаренгейта в температуру за шкалою Цельсія. Скрін коду програми та результати її виконання представити у звіті.

```
 /* Celsius and Fahrengeit */
/* C=(5/9)*(F-32) */
#include<stdio.h>
#include<conio.h>
main()
{
  int fahr,celsius;
  int lower,upper,step;
  lower=0;
  upper=300;
  step=20;
  fahr=lower;
```

```
 printf("\n\nCelsius Fahrengeit\n");
 while( fahr <= upper )
 { celsius = 5*(fahr-32)/9;
   printf("%10d\t%8d\n",fahr,celsius);
   fahr=fahr+step;
   }
 getch();
}
```

4. Скласти програму для створення прямокутного трикутника із зірочок (*), при цьому трикутник має розміри: n рядків у висоту та n символів у ширину. Значення n вводиться з клавіатури. Скрін коду програми та результати її виконання представити у звіті.

5. Оформити звіт.

6. Обчислити скільки зерен необхідно було би видати винахідникові шахів, якщо за першу клітину шахівниці він попросив видати одну зернину пшениці, а за кожну наступну вдвічі більше за попередні. У шахівниці 64 клітини.

7. Для цілих чисел від 1 до 20 обчислити квадратні, кубічні та корені четвертого порядку. Результати звести у таблицю, використовуючи форматування функції printf().

8. Здійснити табулювання функції, що з певними припущеннями з достатньою точністю моделює імпульс Максвела, який утворюється при ударному збудженні широкосмугової антени. Обчислення провести на проміжку зміни **i** в межах [0-31] з кроком i=1, N=32. Результати вивести у вигляді таблиці. Визначити найбільше та найменше значення функції на цьому проміжку.

$$y = i^2 e^{-i^2/100} \sin(\frac{2\pi}{N} i)$$

9. В обчислювальних задачах при програмуванні ітераційних алгоритмів, що закінчуються при досягненні заданої точності, часто необхідна оцінка «машинного нуля», тобто числового значення, менше за яке неможливо задати точність даного алгоритму. Абсолютне значення «машинного нуля» залежить від розрядної сітки застосовуваного комп’ютера, від прийнятої в конкретному трансляторі точності представлення дійсних чисел і від значень, що використовуються для оцінки точності. Наступна програма оцінює абсолютне значення «машинного нуля» відносно близьких (за модулем) до одиниці змінних типу **float**.

```
#include<stdio.h>
#include<conio.h>
```

```
#include<math.h>
void main(void) // Otsinkamashynnohonulia
{
inti=0;                    // i – лічильник ітерацій
 float precision,a;          //  a – допоміжна змінна
precision=1.0;              //  precision – обчислювана точність відносно числа 1.0
m: precision=precision/2.;
  a=precision+1.0;
i++;
  if (a>1.0) goto m;
printf("\n число ділень на 2: %6d\n",i);
printf("машинний нуль: %e\n ",precision);
}
```

Завдання: змінити програмузастосувавши кожного разу один із трьох циклічних операторів.

Оцінку «машинного нуля» провести також для даних типу **double** -формат виведення %le**, longdouble**формат виведення %Le.

10. Обчислити значення скінченної суми, або добутку згідно свого варіанту. Врахувати, що навіть для невеликих чисел значення факторіала може вийти за гранично допустимі для даного типу даних. Аргумент тригонометричних функцій задавати в межах:

$$0 \le X \le \pi / 2 .$$

N=10.

B.1 Дано натуральне число*N*. Обчислити:

$$P = \prod_{i=1}^{N} \left(1 + \frac{1}{\sqrt[4]{i}} \right).$$

B.2 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N} \prod_{j=1}^{i} \frac{j!}{i!}.$$

B.3 Дано натуральне число *N*. Обчислити

$$S = \sum_{k=1}^{N} k^2 \ln(k!).$$

B.4 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N} \frac{i!}{(N+i)!}$$

B.5 Дано натуральні числа *N* і*M=5*. Обчислити

$$F = \frac{M!+N!}{(M+N)!}.$$

В.6 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N}\sum_{j=1}^{i} \sin(0.1 \cdot i + 0.2 \cdot j).$$

В.7 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N}\sum_{k=0}^{i} (i+k)^2.$$

В.8 Дано натуральне число *N*, Обчислити

$$S = \sum_{k=1}^{N} \frac{(-1)^{k+1}}{k(k+1)}.$$

В.9 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N} \frac{x_i}{1+y_i}$$

при $x_1=y_1=1$; $x_{i+1}=0.5*x_i$; $y_{i+1}=y_i+x_i$, i=1,2,…N-1...

В.10 Дано натуральне число *N* ідійсне *x*. Обчислити

$$S_1 = \sum_{i=1}^{N} (\sin x)^i; \; S_2 = \sum_{i=1}^{N} \sin x^i.$$

В.11 Дано натуральне число *N* і дійсне *x=0.1*. Обчислити

$$S = \sum_{i=1}^{N} \frac{1}{x^i}; \; P = \prod_{i=0}^{N} (x-i).$$

В.12 Дано натуральне число *N*. Обчислити

$$S = \sum_{k=1}^{N}\prod_{m=1}^{2k} \sin \frac{mx}{2k+1}.$$

В.13 Дано натуральне число *N*. Обчислити

$$P = \prod_{k=1}^{N} \left| 1 - \frac{k^2}{2k+1} \right|.$$

В.14 Дано натуральне число *N*> 2. Обчислити

$$S = \sum_{k=2}^{N} k^2 \ln(1+k^2).$$

В.15 Дано натуральне число *N*. Обчислити

$$S = \sum_{i=1}^{N}\sum_{k=0}^{i} \frac{i-k}{i+k}.$$

11. Відомо, що одним із методів обчислення багатьох функцій є розкладання їх в ряд Тейлора:

$$y = \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots,$$

$$y = \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + (-1)^n \frac{x^{2n}}{2n!} + \ldots,$$

$$y = e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots,$$

$$y = \ln(1+x) = x - \frac{x^2}{2} + \ldots + (-1)^{n+1} \frac{x^n}{n} + \ldots,$$

$$y = sh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + \frac{x^{2n+1}}{(2n+1)!} + \ldots +,$$

$$y = ch(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{2n!} + \ldots +,$$

Завдання: для заданого **x**, яке уводиться з клавіатури під час роботи програми, обчислити значення функції **y**за допомогою бібліотечних функцій компілятора (наприклад e^x – pow(e,2)), а такожза допомогою вище наведеного явного розкладу її в ряд (ітераційний процес до досягнення заданої точності). Обчислити при цьому також кількість ітерацій, тобто кількість членів ряду в розкладі функції. Точність обчислень, тобто значення члена ряду розкладу функції коли необхідно припиняти ітераційний процес, яке в кожній ітерації обчислюється як різниця між бібліотечним значенням і членом розкладу, **a=0.00001**. Аргумент тригонометричних функцій задавати в межах: $0 \le X \le \pi / 2$.

Приклад 1

```
#include <stdio.h>
int main (void)
{
   int x;
   for (x = 1; x <= 100; x ++)
   printf ( "% d", x);
   return 0;
}
```

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 4

8 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92

93 94 95 96 97 98 99 100

Приклад 2

```
#include<stdio.h>
main()
{
   int i=2;
   while (i<=1024)
{
   i = i*2;
   printf("%d\n",i);
}
}
```

```
 4
 8
 16
 32
 64
 128
 256
 512
 1024
 2048
```

Приклад 3

```
#include<stdio.h>
main()
{
   int j=0,k=0;
   while(j<5)
{
   k+=10;
   j++;
}
   printf("k=%d\n",k);
}
```

```
 k=50
```

Приклад 4

```
#include<stdio.h>
```

```
main()
{
   int a=2, b=10,c;
   do
{
   b=b+a; c=10*a; a++;
}
   while(a<5);
   printf("\nb=%d c=%d a=%d",b,c,a);
}
```

```
b=19 c=40 a=5
```

Приклад 5

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int main(void) {
   int magic;
   int guess;
   srand(time(NULL)); // Ініціалізація генератора випадкових чисел
   magic = rand() % 3; // Генерація магічного числа в діапазоні від 0 до 2
   printf("Вгадайте число: ");
   scanf("%d", &guess);

   if (guess == magic) {
      printf("** Вірно **\n");
      printf("Магічне число рівне %d\n", magic);
   } else {
      printf("Невірно\n");
      while (guess != magic) {
         printf("Вгадайте число: ");
         scanf("%d", &guess);
         printf("Невірно\n");
      }
   }
   return 0;
}
```

```
Вгадайте число: 2
```

Невірно
Вгадайте число: 0
Невірно

Приклад 6

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
   int magic;
   int guess;
   int m = 1;

   srand(time(NULL)); // Ініціалізація генератора випадкових чисел
   magic = rand() % 3;

   printf("Вгадайте число: ");
   scanf("%d", &guess);
   if (guess == magic) {
      printf("** Вірно **\n");
      printf("Магічне число рівне %d\n", magic);
   } else {
      printf("Невірно\n");
      printf("Спроба №%d\n", m);

      while (guess != magic) {
         m++;
         printf("Вгадайте число: ");
         scanf("%d", &guess);
         printf("Невірно\n");
         printf("Спроба №%d\n", m);
      }
   }
   return 0;
}
```

Вгадайте число: 0
Невірно
Спроба №1

Вгадайте число: 1
Невірно
Спроба №2
Вгадайте число: 2
Невірно
Спроба №3

Приклад 7

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
int magic;
int guess;
int m=1; /*кількість спроб*/
srand(time(NULL)); // Ініціалізація генератора випадкових чисел
magic = rand() % 3;
printf("vgaday chuslo:");
scanf("%d",&guess);
if(guess==magic){
printf("**virno**");
printf("magichne chuslo rivne %d", magic);
}
else {
printf("nevirno\n");
printf("m=%d\n",m);
while (guess!=magic){
m++;
printf("vgaday chuslo:");
scanf("%d", &guess);
printf("nevirno\n");
printf("m=%d\n",m);
if(m>9){
printf("magic=%d\n", magic);
break;
}
}
}
}
```

---

vgaday chuslo:1

virnomagichne chuslo rivne 1

Приклад 8

```
/* Celsius and Fahrengeit */
/* C=(5/9)*(F-32) */
#include<stdio.h>
#include<conio.h>
main()
{
   int fahr,celsius;
   int lower,upper,step;
   lower=0;
   upper=300;
   step=20;
   fahr=lower;
   printf("\n\nCelsius Fahrengeit\n");
   while( fahr <= upper )
{ celsius = 5*(fahr-32)/9;
   printf("%10d\t%8d\n",fahr,celsius);
   fahr=fahr+step;
}
   getch();
}
```

```
Celsius Fahrengeit
        0         -17
       20          -6
       40           4
       60          15
       80          26
      100          37
      120          48
      140          60
      160          71
      180          82
      200          93
      220         104
      240         115
      260         126
      280         137
      300         148
```

Приклад 9

```
#include <stdio.h>

int main() {
   int n, i, j;
   printf("Введіть розмір трикутника (кількість рядків): ");
   scanf("%d", &n);
   for (i = 0; i < n; i++) {
      for (j = 0; j <= i; j++) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

```
Введіть розмір трикутника (кількість рядків): 8
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
```

Приклад 10

```
#include <stdio.h>
int main() {
   unsigned  long total_grains = 0; // Загальна кількість зерен
   unsigned  long grains = 1; // Кількість зерен для поточної клітини
   int squares = 64; // Кількість клітин на шахівниці

   // Обчислення кількості зерен для кожної клітини та загальної кількості
   for (int i = 1; i <= squares; i++) {
      total_grains += grains;
      grains *= 2; // Подвоєння кількості зерен для наступної клітини
   }

   // Виведення результату
   printf("Загальна кількість зерен, що потрібно видати: %llu\n", total_grains);
```

```
    return 0;
}
```

Загальна кількість зерен, що потрібно видати: 18446744073709551615

Приклад 11

```
#include <stdio.h>
#include <math.h>
int main() {
    int i;
    printf("Число\tКвадрат\tКуб\tКорінь 4-го степеня\n");
    for (i = 1; i <= 20; i++) {
        printf("%d\t%d\t%d\t%.2f\n", i, i * i, i * i * i, pow(i, 0.25));
    }}
```

Число	Квадрат	Куб	Корінь 4-го степеня
1	1	1	1.00
2	4	8	1.19
3	9	27	1.32
4	16	64	1.41
5	25	125	1.50
6	36	216	1.57
7	49	343	1.63
8	64	512	1.68
9	81	729	1.73
10	100	1000	1.78
11	121	1331	1.82
12	144	1728	1.86
13	169	2197	1.90
14	196	2744	1.93
15	225	3375	1.97
16	256	4096	2.00
17	289	4913	2.03
18	324	5832	2.06
19	361	6859	2.09
20	400	8000	2.11

Приклад 12

```
#include <stdio.h>
#include <math.h>
```

```
#define PI 3.14159265
#define EXP 2.71828182
int main(void) {
    float y;
    int N = 32;
    float a = INFINITY;     // Початкове значення для мінімуму
    float b = -INFINITY;    // Початкове значення для максимуму
    float res[N];           // Масив для зберігання значень функції
    for (int i = 0; i < N; i++) {
        y = pow(i, 2) * pow(EXP, (-pow(i, 2) / 100.0)) * sin((2 * PI / N) * i);
        res[i] = y;
        b = fmax(b, y);     // Використовуємо fmax для визначення максимуму
        a = fmin(a, y);     // Використовуємо fmin для визначення мінімуму
        printf("i = %d, y = %f\n", i, y);
    }
    printf("max = %f\n", b);
    printf("min = %f\n", a);
    return 0;
}
```

```
// i = 0, y = 0.000000
// i = 1, y = 0.193149
// i = 2, y = 1.470713
// i = 3, y = 4.569777
// i = 4, y = 9.640906
// i = 5, y = 16.188730
// i = 6, y = 23.204479
// i = 7, y = 29.441893
// i = 8, y = 33.746716
// i = 9, y = 35.341129
// i = 10, y = 33.987629
// i = 11, y = 30.000978
// i = 12, y = 24.124784
// i = 13, y = 17.324791
// i = 14, y = 10.565220
// i = 15, y = 4.626533
// i = 16, y = 0.000000
// i = 17, y = -3.133448
// i = 18, y = -4.855909
// i = 19, y = -5.425541
// i = 20, y = -5.180445
```

```
 i = 21, y = -4.457038
 i = 22, y = -3.535700
 i = 23, y = -2.615844
 i = 24, y = -1.815040
 i = 25, y = -1.183351
 i = 26, y = -0.723988
 i = 27, y = -0.413587
 i = 28, y = -0.218239
 i = 29, y = -0.104020
 i = 30, y = -0.042504
 i = 31, y = -0.012572
 max = 35.341129
min = -5.425541
```

Приклад 13

```
//FLOAT
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
float precision = 1.0, a = 1.0 + precision;
for (precision = 1.0; a > 1.0; ++i) {
precision = precision / 2;
a = 1.0 + precision;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %e\n", precision);
}
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
float precision = 1.0;
float a = 1.0 + precision;
while (a > 1.0) {
precision = precision / 2;
a = 1.0 + precision;
++i;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %e\n", precision);
}
#include<stdio.h>
```

```
#include<math.h>
void main(void){
int i=0;
float precision,a;
precision = 1.0;
do{
precision = precision/2;
a = 1.0 + precision;
++i;}
while(a>1);
printf("\nчисло ділень на 2: %6d\n",i);
printf("машинний нуль: %e\n ",precision);
}
```

```
Число ділень на 2:     24
Машинний нуль: 5.960464e-08
```

```
//double
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
double precision = 1.0, a = 1.0 + precision;
for (precision = 1.0; a > 1.0; ++i) {
precision = precision / 2;
a = 1.0 + precision;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %e\n", precision);
}
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
double precision = 1.0;
double a = 1.0 + precision;
while (a > 1.0) {
precision = precision / 2;
a = 1.0 + precision;
++i;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %e\n", precision);
```

```
}
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
double precision, a;
precision = 1.0;
do {
precision = precision / 2;
a = 1.0 + precision;
++i;
} while (a > 1.0);
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %e\n", precision);
}
```

```
Число ділень на 2:     53
Машинний нуль: 1.110223e-16
```

```
//long double
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
long double precision = 1.0L, a = 1.0L + precision;
for (precision = 1.0L; a > 1.0L; ++i) {
precision = precision / 2;
a = 1.0L + precision;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %Le\n", precision);
}
```

```
#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
long double precision = 1.0L;
long double a = 1.0L + precision;
while (a > 1.0L) {
precision = precision / 2;
a = 1.0L + precision;
```

```
++i;}
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %Le\n", precision);
}

#include <stdio.h>
#include <math.h>
int main(void) {
int i = 0;
long double precision, a;
precision = 1.0L;
do {
precision = precision / 2;
a = 1.0L + precision;
++i;
} while (a > 1.0L);
printf("\nЧисло ділень на 2: %6d\n", i);
printf("Машинний нуль: %Le\n", precision);
}
```

```
 Число ділень на 2:     64
 Машинний нуль: 5.421011e-20
```

Завдання

```
#include <stdio.h>
#include <math.h>
#define N 10

int main() {
    double sum = 0.0;
    for (int i = 1; i < N; i++) {
        for (int j = 1; j <= i; j++) {
            sum += sin(0.1 * i + 0.2 * j);
        }
    }
    printf("Значення суми: %lf\n", sum);
    return 0;
}
```

```
Значення суми: 36.928994
```

Завдання

```
#include <stdio.h>
```

```
#include <math.h>

int main() {
   double x, y_library, y_series, term, a;
   int iterations = 0;
   const double precision = 0.00001;

   printf("Введіть значення x (0 <= x <= pi/2): ");
   scanf("%lf", &x);

   y_library = exp(x);  // Виправлено обчислення за допомогою бібліотечної функції
   y_series = 1.0;  // Початкове значення ряду для exp(x)
   a = 1.0;
   term = x;

   while (fabs(term) >= precision) {
      y_series += term;
      iterations++;
      a *= iterations;  // Факторіал
      term = pow(x, iterations) / a;
   }

   printf("Значення функції y за допомогою бібліотечної функції: %f\n", y_library);
   printf("Значення функції y за допомогою ряду: %f\n", y_series);
   printf("Кількість ітерацій: %d\n", iterations);

   return 0;
}
```

```
Введіть значення x (0 <= x <= pi/2): 3
Значення функції y за допомогою бібліотечної функції: 20.085537
Значення функції y за допомогою ряду: 23.085534
Кількість ітерацій: 16
```

Контрольні питання

1. Призначення операторів циклу.
   Призначення операторів циклу полягає в тому, щоб виконувати певний блок коду або набір інструкцій декілька разів, доки виконуються певні умови.
2. Конструкція оператора циклу while.

Оператор циклу `while` виконується, доки задана умова є істинною. Спочатку перевіряється умова, і якщо вона істинна, виконується тіло циклу.

3. Конструкція оператора циклу do- while.
   Оператор циклу `do-while` подібний до `while`, але умова перевіряється після виконання тіла циклу. Таким чином, тіло циклу виконується принаймні один раз, навіть якщо умова вже невірна.

4. Конструкція оператора циклу for.

   . Оператор циклу `for` використовується для повторення блоку коду певне число разів. Він складається з трьох основних частин: ініціалізація, умова і крок. Цей цикл часто використовується, коли заздалегідь відомо, скільки разів потрібно виконати цикл.

5. Поясніть призначення виразів у конструкції циклу for.
   У конструкції циклу `for` вирази мають такі призначення:
   - Ініціалізаційний вираз: виконується один раз на початку циклу і встановлює початкові значення змінних.
   - Умова: перевіряється перед кожним виконанням тіла циклу і вказує, чи має цикл продовжуватися.
   - Крок: виконується після кожного виконання тіла циклу і використовується для зміни значень змінних керування циклом.

Висновок: під чпс виконання цієї лабораторної роботи я ознайомився з особливостями функціонування операторів циклу та навчився їх використовувати у процесі програмування.