**МИНЕСТРЕСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Балтийский государственный технический университет «ВОЕНМЕХ»

им. Д.Ф. Устинова»

Кафедра О7

«Информационные системы и программная инженерия»

Практическое задание № 3

по дисциплине «Информационные технологии и программирование»

на тему «Программирование циклов»

Выполнила:

Студент(ка) Карпинская Полина Анатольевна

Группа Е721б

Преподаватель: Удовиченко Андрей Сергеевич,

канд., физ.-мат. наук., доцент

Санкт-Петербург

2023

**Цель работы** – научиться программировать циклические алгоритмы с помощью операторов for, while и do…while.

Скрипт practical\_task\_3\_1.c:

#include <stdio.h>

int main() {

int num;

int sum;

int square;

printf("two-digit numbers that meet the condition\n");

for (num = 10; num < 100; num++) {

sum = (num / 10) + (num % 10);

square = sum \* sum;

if (num == sum + square) printf("%d\n", num);

}

return 0;

}

Скрипт practical\_task\_3\_2.c:

#include <stdio.h>

#include <math.h>

int main() {

float A, B;

int m = 1;

printf("Enter numbers A and B (A && B > 1):\n");

if (scanf("%f%f", &A, &B) == 2 && getchar() == '\n') {

while (pow(B, m) <= m \* A) m++;

printf("The smallest natural number m that satisfies the condition B^m > m \* A: %d\n", m);

} else printf("Error! Incorrect values entered!");

return 0;

}

Скрипт practical\_task\_3\_3.c:

#include <stdio.h>

#include <math.h>

#include <conio.h>

int factorial(int n) {

int sum = n;

if (n > 1) sum = sum \* factorial(n - 1);

return sum;

}

double calculateSum(double x) {

double previous, current;

double sum = 0.0;

int n = 1;

current = (pow(-1, n)\*pow(x, n-1))/(factorial(n+1));

sum += current;

n++;

do {

previous = current;

current = (pow(-1, n)\*pow(x, n-1))/(factorial(n+1));

sum += current;

n++;

} while (abs(current - previous) > pow(10, -6));

return sum;

}

int main() {

double x;

if (scanf("%lf", &x) == 1 && x >= -1 && x <= 1 && getchar() == '\n') {

double res = calculateSum(x);

printf("Amount with an accuracy of 10^-6: %lf\n", res);

} else printf("Error! Incorrect value entered!");

return 0;

}

Ответы на контрольные вопросы:

1. Цикл - это конструкция в программировании, которая позволяет выполнять определенный блок кода множество раз до тех пор, пока выполняется определенное условие.

2. Виды циклов:

- цикл с предусловием (while)

- цикл с постусловием (do-while)

- цикл с параметром (for)

3. Цикл с предусловием выполняет тело цикла только тогда, когда условие истинно, в то время как цикл с постусловием выполняет тело цикла хотя бы один раз и затем проверяет условие.

4. Цикл с предусловием используется, когда условие выполнения цикла определяется перед его выполнением, а цикл с постусловием используется, когда нужно выполнить тело цикла хотя бы один раз перед проверкой условия. Например, цикл с предусловием можно использовать для обхода элементов массива, пока индекс меньше размера массива, а цикл с постусловием может использоваться для ввода данных пользователем, где нужно выполнить ввод хотя бы один раз.

5. Цикл с предусловием в языке Си - while.

6. Тело цикла с предусловием содержит от одного до нескольких операторов.

7. Цикл с постусловием на языке Си записывается следующим образом:

do {

// тело цикла

} while (условие);

8. Бесконечный цикл можно задать, например, с помощью цикла while, где условие всегда истинно. Он может использоваться, когда нужно выполнять определенные действия до тех пор, пока не будет выполнено какое-либо условие для выхода. Чтобы выйти из бесконечного цикла, можно использовать оператор break внутри цикла или изменить условие выхода с помощью оператора if.

9. Значение выражения, определяющего условие выполнения цикла, должно быть ложным (false), чтобы завершить цикл.

10. Значение выражения, определяющего условие выполнения цикла, должно быть истинным (true), чтобы выполнить тело цикла.

11. Неправильное задание выражения, определяющего условие выполнения цикла, может привести к бесконечному циклу или неправильным результатам. Например, если условие всегда истинно, цикл не завершится.

12. Тело цикла может отсутствовать, если используется цикл с предусловием while и условие истинно изначально. Например:

while (true) {

// тела цикла нет

}

13. Оператор while и оператор if похожи в том, что они оба проверяют условие и выполняют определенный блок кода, если условие истинно. Однако, оператор while используется для организации циклов, тогда как оператор if используется для выполнения условного блока кода один раз.

14. Порядок действий при выполнении цикла for следующий:

- Выполняется инициализация переменных перед началом цикла.

- Проверяется условие цикла.

- Если условие истинно, выполняется тело цикла.

- Обновляются переменные в блоке обновления.

- Возвращаемся к проверке условия. Если условие по-прежнему истинно, цикл повторяется. Если условие ложно, цикл завершается.

15. Чтобы организовать арифметический цикл с помощью цикла for, необходимо:

- Задать начальное значение переменной, от которой будет осуществляться счет.

- Задать условие, при котором цикл будет выполняться.

- Задать инструкцию для обновления значения переменной после каждой итерации.

Пример:

for (int i = 0; i < 10; i++) {

// вычисления

}

16. Алгоритм, определяемый циклом for, может быть записан с помощью цикла while следующим образом:

int i = 0;

while (i < 10) {

// вычисления

i++;

}

17. Вложенный цикл - это цикл, который находится внутри другого цикла. То есть, один цикл включает в себя другой цикл.

18. Количество выполнений тела вложенного цикла зависит от условий, но обычно тело вложенного цикла выполняется несколько раз.

19. Оператор break используется для прекращения выполнения цикла внутри блока, в котором он используется. Оператор continue используется для пропуска текущей итерации цикла и перехода к следующей итерации.

20. Рекуррентные вычисления - это вычисления, основанные на повторном применении тех же вычислений к ранее полученным результатам. Они используются, например, для решения задач, для которых требуется постепенное накопление результатов. Программирование рекуррентных вычислений предполагает организацию функций, вызывающих сами себя с изменяющимися параметрами, для достижения поставленной цели или получения результата.