Лабораторная работа 2.
Разветвляющиеся программы. Циклы

1. Цель работы.

1) Изучение базовых конструкций языка C++.
2) Получение навыков программирования программ на C++ с ветвлениями и циклами.

2. Теоретическая часть. Базовые конструкции структурного программирования

Следование, ветвление и цикл называют *базовыми конструкциями* структурного программирования. *Следованием* называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных). *Ветвление* задает выполнение либо одного, либо другого оператора в зависимости от выполнения какого-либо условия. *Цикл* задает многократное выполнение оператора.

В большинстве языков высокого уровня существует несколько реализаций базовых конструкций; в C++ есть три вида циклов и два вида ветвлений (на два и на произвольное количество направлений).

Рассмотрим операторы языка, реализующие базовые конструкции структурного программирования.

Условный оператор `if`

Условный оператор `if` используется для разветвления процесса вычислений на два направления. Формат оператора:

```
if ( выражение ) оператор_1; [else оператор_2;]
```

Сначала вычисляется выражение, которое может иметь арифметический тип или тип указателя. Если оно не равно нулю (имеет значение `true`), выполняется первый оператор, иначе – второй. После этого управление передается на оператор, следующий за условным. Одна из ветвей может отсутствовать.

Оператор `switch`

Оператор `switch` (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Формат оператора:

```
switch ( выражение ){
   case константное_выражение_1: [список_операторов_1]
   case константное_выражение_2: [список_операторов_2]
   …
   case константное_выражение_n: [список_операторов_n]
   [default: операторы ]
}
```

Выполнение оператора начинается с вычисления выражения (оно должно быть целочисленным), а затем управление передается первому оператору из списка, помеченного константным выражением, значение которого совпало с вычисленным. *После этого, если выход из переключателя явно не указан, последовательно выполняются все остальные ветви.*

Выход из переключателя обычно выполняется с помощью операторов `break` или `return`. Оператор `break` выполняет выход из самого внутреннего из объемлющих его

операторов `switch`, `for`, `while` и `do`. Оператор `return` выполняет выход из функции, в теле которой он записан.

Все константные выражения должны иметь разные значения, но быть одного и того же целочисленного типа. Несколько меток могут следовать подряд. Если совпадения не произошло, выполняются операторы, расположенные после слова `default` (а при его отсутствии управление передается следующему за `switch` оператору).

Хотя наличие слова `default` и не обязательно, рекомендуется всегда обрабатывать случай, когда значение выражения не совпадает ни с одной из констант. Это облегчает поиск ошибок при отладке программы.

Оператор `switch` предпочтительнее оператора `if` в тех случаях, если в программе требуется разветвить вычисления на количество направлений, большее двух, и выражение, по значению которого производится переход на ту или иную ветвь, является целочисленным. Часто это справедливо даже для двух ветвей, поскольку улучшает наглядность программы.

Цикл с предусловием (`while`)

Цикл с предусловием имеет вид:

```
while ( выражение ) оператор;
```

Выражение определяет условие повторения тела цикла, представленного простым или составным оператором. Выполнение оператора начинается с вычисления выражения. Если оно истинно (не равно `false`), выполняется оператор цикла. Если при первой проверке выражение равно `false`, цикл не выполнится ни разу. Тип выражения должен быть арифметическим или приводимым к нему. Выражение вычисляется перед каждой итерацией цикла.

Распространенный прием программирования – организация бесконечного цикла с заголовком `while (true)` либо `while (1)` и принудительным выходом из тела цикла по выполнению какого-либо условия.

В круглых скобках после ключевого слова `while` можно вводить описание переменной. Областью ее действия является цикл:

```
while (int x = 0){…/* область действия x */}
```

Цикл с постусловием (`do while`)

Цикл с постусловием имеет вид:

```
do оператор while (выражение);
```

Сначала выполняется простой или составной оператор, составляющий тело цикла, а затем вычисляется выражение. Если оно истинно (не равно `false`), тело цикла выполняется еще раз. Цикл завершается, когда выражение станет равным `false` или в теле цикла будет выполнен какой-либо оператор передачи управления. Тип выражения должен быть арифметическим или приводимым к нему.

Цикл с параметром (`for`)

Цикл с параметром имеет следующий формат:

```
for (инициализация; выражение; модификации) оператор;
```

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. В этой части можно записать несколько операторов, разделенных запятой (операцией «последовательное выполнение»), например, так:

```
for (int i = 0, j = 2; ...
int k, m;
for (k = 1, m = 0; ...
```

Областью действия переменных, объявленных в части инициализации цикла, является цикл. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат, приведенный к типу `bool`, равен `true`, цикл выполняется. Цикл с параметром реализован как цикл с предусловием.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла. В части модификаций можно записать несколько операторов через запятую. Простой или составной *оператор* представляет собой тело цикла. Любая из частей оператора `for` может быть опущена (но точки с запятой надо оставить на своих местах!).

Любой цикл `while` может быть приведен к эквивалентному ему циклу `for` и наоборот по следующей схеме:

```
for (b1; b2; b3) оператор;
```

```
b1;
while (b2){
   оператор; b3;}
```

Операторы цикла взаимозаменяемы, но можно привести некоторые *рекомендации* по выбору наилучшего в каждом конкретном случае.

Оператор `do while` обычно используют, когда цикл требуется обязательно выполнить хотя бы раз (например, если в цикле производится ввод данных).

Оператором `while` удобнее пользоваться в случаях, когда число итераций заранее не известно, очевидных параметров цикла нет или модификацию параметров удобнее записывать не в конце тела цикла.

Оператор `for` предпочтительнее в большинстве остальных случаев (однозначно – для организации циклов со счетчиками).

Оператор `break`

Оператор `break` используется внутри операторов цикла или `switch` для обеспечения перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится `break`.

Оператор `continue`

Оператор перехода к следующей итерации цикла `continue` пропускает все операторы, оставшиеся до конца тела цикла, и передает управление на начало следующей итерации.

3. Практическая часть

Вычисление значения функции, заданной графически

Требуется написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика (рис. 1).

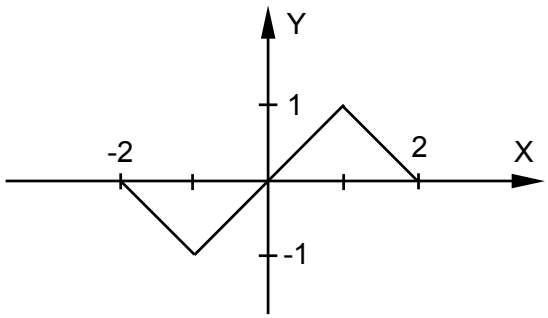

Рис. 1. Функция для задачи

Исходными данными является вещественное значение аргумента x, который определен на всей числовой оси, а результатом – вещественное значение функции y.

Перед написанием программы следует составить алгоритм ее решения. Алгоритм можно описать в различном виде, например, в словесном или в виде блок-схемы. Для начала, удобнее будет записать функцию в виде формул:

$$y = \begin{cases} 0, & x < -2 \\ -x - 2, & -2 \le x < -1 \\ x, & -1 \le x < 1 \\ -x + 2, & 1 \le x < 2 \\ 0, & x \ge 2 \end{cases}$$

Ниже приведено описание алгоритма в неформальной словесной форме. Этот способ рекомендуется потому, что только после четкого описания задачи на естественном языке, ее можно успешно записать на языке программирования.

1. Ввести значение аргумента x.
2. Определить, какому интервалу из области определения функции оно принадлежит.
3. Вычислить значение функции y по соответствующей формуле.
4. Вывести значение y.

Первый вариант программы будет выглядеть так:

```
#include <iostream.h>
int main(){
   float x, y;
   cout << " Введите значение аргумента" << endl;
   cin >> x;
   if ( x <  -2 )           y = 0;
   if ( x >= -2 && x < -1 ) y = -x - 2;
   if ( x >= -1 && x <  1 ) y = x;
   if ( x >=  1 && x <  2 ) y = -x + 2;
   if ( x >=  2 )           y = 0;
   cout << " Для x = " << x << " значение функции y = " << y << endl;
   return 0;
}
```

Тестовые примеры для этой программы должны включать по крайней мере по одному значению аргумента из каждого интервала, а для проверки граничных условий – еще и все точки перегиба (если это кажется вам излишним, попробуйте в последнем условии «забыть» знак =, а затем ввести значение x, равное 2).

Операции отношения (<, >, ==, ≤, ≥, !=) являются бинарными, то есть имеют два операнда, и формируют результат типа `bool`, равный `true` или `false`. Поскольку необходимо, чтобы эти условия выполнялись одновременно, они объединены с помощью операции логического И (`&&`). Приоритет у операции И ниже, чем у операций отношения, поэтому заключать их в скобки не требуется.

Напишите код программы в редакторе среды программирования и запустите ее на выполнение.

Напишите вариант программы с использованием функций ввода-вывода в стиле C и запустите ее на выполнение:

```
#include <stdio.h>
int main(){
   float x, y;
   printf(" Введите значение аргумента: \n");
   scanf("%f", &x);
   if ( x < -2 )    y = 0;
   if ( x >= -2 && x < -1 ) y = -x - 2;
   if ( x >= -1 && x <  1 ) y = x;
   if ( x >=  1 && x <  2 ) y = -x + 2;
   if ( x >=  2 )   y = 0;
   printf(" Для x = %5.2f значение функции y = %5.2f\n", x, y);
   return 0;
}
```

Определение попадания точки в заданную область

Дана заштрихованная область (рис. 2) и точка с координатами (x, y). Написать программу, определяющую, попадает ли точка в область. Результат вывести в виде текстового сообщения.

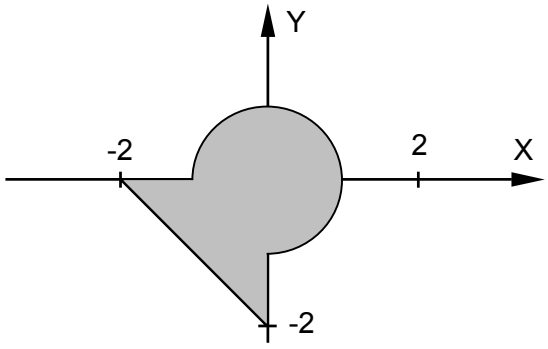

Рис. 2. Графически заданная область для задачи

Запишем условия попадания точки в область в виде формул. Область можно описать как круг, пересекающийся с треугольником. Точка может попадать либо в круг, либо в треугольник, либо в их общую часть:

$$\{x^2 + y^2 \le 1\} \quad \text{или} \quad \begin{cases} x \le 0 \\ y \le 0 \\ y \ge -x - 2 \end{cases}$$

Первое условие задает попадание точки в круг, второе – в треугольник. Программа для решения задачи выглядит следующим образом:

```
#include <iostream.h>
int main(){
   float x, y;
   cout << " Введите значения x и y: " << endl;
   cin >> x >> y;
   if (x*x + y*y <= 1 || x <= 0 && y <= 0 && y >= - x - 2)
      cout << " Точка попадает в область" << endl;
   else cout << " Точка не попадает в область" << endl;
   return 0;
}
```

Три условия из правых фигурных скобок должны выполняться одновременно, поэтому в программе они объединяются с помощью операции И. Ее приоритет выше, чем у ИЛИ (||), и ниже, чем у операций отношения, поэтому дополнительных скобок не требуется.

Выполните программу несколько раз, задавая различные положения точки. Задайте заштрихованную область какого-либо другого вида и измените программу в соответствии с этой областью.

ВНИМАНИЕ. Следует избегать проверки вещественных величин на равенство; вместо этого лучше сравнивать модуль их разности с некоторым малым числом. Это связано с погрешностью представления вещественных значений в памяти:

```
float a, b;
if (a == b) cout << " равны";
   else cout << " не равны";            /* Не рекомендуется! */
if (fabs(a - b) < 1e-6 ) cout << " равны";
   else cout << " не равны";            // Верно!
```

Значение величины, с которой сравнивается модуль разности, следует выбирать в зависимости от решаемой задачи и точности участвующих в выражении переменных. Снизу эта величина ограничена определенными в заголовочном файле `<float.h>` константами `FLT_EPSILON = 1.192092896e-07F` и `DBL_EPSILON = 2. 2204460492503131e-016`. (`FLT_EPSILON` – это минимально возможное значение переменной типа `float`, такое, что `1.0 + FLT_EPSILON != 1.0`, `DBL_EPSILON` – аналогичная константа для типа `double`).

Для присваивания какой-либо переменной в зависимости от выполнения условия двух различных значений лучше пользоваться не оператором if, а тренарной условной операцией, например:

```
if (a < b) c = x; else c = y;       // Нерационально
c = (a < b)? x: y;                  // Рекомендуется
```

Тернарной эта операция называется потому, что у нее три операнда. Первый операнд представляет собой выражение, результат вычисления которого преобразуется в значение `true` или `false`. После знака вопроса через двоеточие записываются два выражения. Результат вычисления первого из них принимается за результат всей операции, если первый операнд имеет значение `true`. В противном случае результатом всей операции является результат вычисления второго выражения. Таким образом, переменной c будет присвоено значение либо переменной `x`, либо `y`.

Пример применения оператора `switch`

Написать программу, определяющую, какая из курсорных клавиш была нажата.

В составе библиотеки, унаследованной от языка C, есть функция `getch()`, возвращающая код нажатой пользователем клавиши. В случае нажатия функциональных или курсорных клавиш эта функция возвращает `0` либо `0xE0` (в зависимости от компилятора), а ее повторный вызов позволяет получить расширенный код клавиши.

```
#include <stdio.h>
#include <conio.h>
int main(){
   int key;
   printf("\n Нажмите одну из курсорных клавиш: \n");
   key = getch(); key = getch();
   switch (key){
      case 77: printf("стрелка вправо\n"); break;
      case 75: printf("стрелка влево\n"); break;
      case 72: printf("стрелка вверх\n"); break;
      case 80: printf("стрелка вниз\n"); break;
      default: printf("не стрелка\n");
   }
   return 0;
}
```

Выражение, стоящее в скобках после ключевого слова `switch`, а также константные выражения в `case` должны быть целочисленного типа (они неявно приводятся к типу выражения в скобках). Если требуется выполнить одни и те же действия при нескольких различных значениях констант, метки перечисляются одна за другой, например:

```
case 77: case 75: case 72: case 80: printf("стрелки"); break;
```

Метки сами по себе не вызывают изменения порядка выполнения операторов, поэтому если вы не хотите, чтобы управление было автоматически передано на первый оператор следующей ветви, необходимо после каждой ветви использовать оператор `break`.

Циклические программы. Расчет и вывод таблицы значений функции

При написании любого цикла надо иметь в виду, что в нем всегда явно или неявно присутствуют четыре элемента: начальные установки, тело цикла, модификация параметра цикла и проверка условия продолжения цикла.

Написать программу печати таблицы значений функции

$$y = \begin{cases} t, & x < 0 \\ tx, & 0 \le x < 10 \\ 2t, & x \ge 10 \end{cases}$$

для аргумента, изменяющегося в заданных пределах с заданным шагом. Если $t > 100$*, должны выводиться целые значения функции*.

Исходными данными являются начальное значение аргумента `Xn`, конечное значение аргумента `Xk`, шаг изменения агрумента `dX` и параметр `t`. Все величины – вещественные. Программа должна выводить таблицу, состоящую из двух столбцов – значений аргумента и соответствующих им значений функции.

В словесной форме алгоритм можно сформулировать так:

1. Ввести исходные данные.
2. Взять первое из значений аргумента.

3. Определить, какому из интервалов оно принадлежит.

4. Вычислить значение функции `y` по соответствующей формуле.

5. Если `t` > 100, преобразовать значение `y` в целое.

6. Вывести строку таблицы.

7. Перейти к следующему значению аргумента.

8. Если оно не превышает конечное значение, повторить шаги 3–7, иначе закончить выполнение.

В каждый момент времени требуется хранить одно значение функции, поэтому для него достаточно завести одну переменную вещественного типа. Шаги 3–7 повторяются многократно, поэтому для их выполнения надо организовать цикл. В приведенном ниже варианте используется цикл `while`:

```
#include <stdio.h>
#include <math.h>
int main(){
   double Xn, Xk, dX, t, y;
   cout << Rus("Введите Xn, Xk, dX, t\n");
   scanf("%lf%lf%lf%lf", &Xn, &Xk, &dX, &t);
   printf(" ------------------- \n");
   printf("|    X    |    Y    |\n");
   printf(" ------------------- \n");
   double x = Xn;                  // Начальные установки
   while (x <= Xk){
      if (x < 0) y = t;
      if (x >= 0 && x < 10) y = t*x;
      if ( x >= 10) y = 2*t;
      if (t > 100) printf("|%9.2lf|%9d|\n", x, (int)y);
      else printf("|%9.2lf|%9.2lf|\n", x, y);
      x += dX;                     // Модификация параметра цикла
   }
   printf(" ------------------- \n");
   return 0;
}
```

Та же программа с использованием оператора `for` будет следующей:

```
#include <stdio.h>
#include <math.h>
int main(){
   double Xn, Xk, dX, t, y;
   cout << Rus("Введите Xn, Xk, dX, t\n");
   scanf("%lf%lf%lf%lf", &Xn, &Xk, &dX, &t);
   printf(" ------------------- \n");
   printf("|    X    |    Y    |\n");
   printf(" ------------------- \n");
   for (double x = Xn; x <= Xk; x += dX){
      if (x < 0) y = t;
      if (x >= 0 && x < 10) y = t*x;
      if ( x >= 10) y = 2*t;
      if (t > 100) printf("|%9.2lf|%9d|\n", x, (int)y);
      else printf("|%9.2lf|%9.2lf|\n", x, y);
   }
   printf(" ------------------- \n");
   return 0;}
```

В программу введена вспомогательная переменная `x`, которая последовательно принимает значения от `Xn` до `Xk` с шагом `dX`. Она определена непосредственно перед использованием. Это снижает вероятность ошибок (например, таких, как использование неинициализированной переменной).

Для преобразования значения функции к целому в программе использовалась конструкция `(int)y`, унаследованная из языка C. Строго говоря, в данном случае лучше применить операцию преобразования типа `static_cast`, введенную в C++.

```
printf("|%9.2lf |%9d |\n", x, static_cast<int>(y));
```

Выполните программу несколько раз, задавая различные значения исходных данных. С помощью ручного просчета убедитесь в правильности вычислений.

Вычисление суммы ряда

Рассмотрим пример цикла, количество итераций которого заранее подсчитать невозможно. *Требуется написать программу вычисления значения функции* `Chx` *(гиперболический косинус) с помощью бесконечного ряда Тейлора с точностью* `ε` *по формуле*:

$$y = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots + \frac{x^{2n}}{2n!} + \ldots$$

Этот ряд сходится при `|x|` $< \infty$. Для достижения заданной точности требуется суммировать члены ряда, абсолютная величина которых больше `ε`. Для сходящегося ряда модуль члена ряда C_n при увеличении `n` стремится к нулю. При некотором `n` неравенство $|C_n| > \varepsilon$ перестает выполняться, и вычисления прекращаются.

Общий алгоритм решения этой задачи очевиден: требуется задать начальное значение суммы ряда, а затем многократно вычислять очередной член ряда и добавлять его к ранее найденной сумме. Вычисления заканчиваются, когда абсолютная величина очередного члена ряда станет меньше заданной точности.

До выполнения программы предсказать, сколько членов ряда потребуется просуммировать, невозможно. В цикле такого рода есть опасность, что он никогда не завершится – как из-за возможных ошибок в вычислениях, так и из-за ограниченной области сходимости ряда (данный ряд сходится на всей числовой оси, но существуют ряды Тейлора, которые сходятся только для определенного интервала значений аргумента). Поэтому для надежности программы необходимо предусмотреть аварийный выход из цикла с печатью предупреждающего сообщения по достижении некоторого максимально допустимого количества итераций.

Прямое вычисление члена ряда по приведенной выше общей формуле, когда `x` возводится в степень, вычисляется факториал, а затем числитель делится на знаменатель, имеет два недостатка, которые делают этот способ непригодным. Первый недостаток – большая погрешность вычислений. При возведении в степень и вычислении факториала можно получить очень большие числа, при делении которых друг на друга произойдет потеря точности, поскольку количество значащих цифр, хранимых в ячейке памяти, ограничено. Второй недостаток связан с эффективностью вычислений: как легко заметить, при вычислении очередного члена ряда нам уже известен предыдущий, поэтому вычислять каждый член ряда «от печки» нерационально.

Для уменьшения количества выполняемых действий следует воспользоваться рекуррентной формулой получения последующего члена ряда через предыдущий $C_{n+1} = C_n T$, где T – некоторый множитель. Подставив в эту формулу C_n и C_{n+1}, получим выражение для вычисления T:

$$T = \frac{C_{n-1}}{C_n} = \frac{2n! \cdot x^{2(n+1)}}{x^{2n} \cdot (2(n+1))!} = \frac{x^2}{(2n+1)(2n+2)}$$

Ниже приведен текст программы с комментариями. Обратите внимание на заголовок цикла: в нем, кроме части начальных установок, заданы условие выхода из цикла и модификация параметра цикла, то есть в данной программе все составные части цикла присутствуют в явном виде.

```
#include <iostream.h>
#include <math.h>
int main(){
   const int MaxIter=500; /*максимально допустимое количество итераций*/
   double x, eps;
   cout << "\nВведите аргумент и точность:"; cin >> x >> eps;
   bool done = true;      // признак достижения точности
   double ch=1, y=ch;     // первый член ряда и нач. значение суммы
   for (int n = 0; fabs(ch) > eps; n++) {
      ch *= x*x/((2*n + 1)*(2*n + 2)); // очередной член ряда
   y += ch;               // добавление члена ряда к сумме
   if (n>MaxIter){
      cout << "\nРяд расходится!";
      done = false; break; }
   }
   if (done){
      cout << "\nЗначение функции: " << y << " для x = " << x << endl;
      cout << "вычислено после " << n << " итераций" << endl;
   }
   return 0;
}
```

Первый член ряда равен 1, поэтому чтобы при первом проходе цикла значение второго члена вычислялось правильно, `n` должно быть равно 0. Максимально допустимое количество итераций удобно задать с помощью именованной константы. Для аварийного выхода из цикла применяется оператор `break`, который выполняет выход на первый после цикла оператор.

Поскольку выход как в случае аварийного, так и в случае нормального завершения цикла происходит в одну и ту же точку программы, вводится булева переменная `done`, которая предотвращает печать значения функции после выхода из цикла в случае, когда точность вычислений не достигнута. Создание подобных переменных-«флагов», принимающих значение «истина» в случае успешного окончания вычислений и «ложь» в противном случае, является распространенным приемом программирования.

Измените программу так, чтобы она печатала не только значения аргумента и функции, но и количество просуммированных членов ряда, и выполните программу несколько раз для различных значений аргумента и точности. Выявите зависимость между этими величинами.

Можно реализовать ту же логику и без специальной булевой переменной, объединив проверку обоих вариантов выхода из цикла в его заголовке:

```
#include <iostream.h>
#include <math.h>
#include <float.h>
int main(){
   const int Maxlter = 500; /* максимально допустимое количество итераций */
   double x, eps = DBL_EPSILON;
   cout << "\nВведите аргумент"; cin >> x;
   double ch = 1, y = ch; // первый член ряда и нач. значение суммы
   for (int n = 0; fabs(ch) > eps && n < MaxIter; n++)
      ch *= x*x /((2*n + 1)*(2*n + 2)); // очередной член ряда
      y += ch;
```

```
    }
    if (n < MaxIter){
       cout << "\nЗначение функции:. " << y << " для x = " << x << endl;
       cout << "вычислено после " << n << " итераций" << endl;
    }
    else cout << "\nРяд расходится!";
    return 0;
}
```

В этом варианте программы сумма ряда для разнообразия вычисляется с максимально возможной точностью.

В библиотеке есть функция `cosh(x)`, вычисляющая гиперболический косинус. Ее прототип находится в файле `<math.h>`. Измените программу так, чтобы она рядом со значением, вычисленным через разложение в ряд, печатала и значение, определенное с помощью стандартной функции. Сравните результаты вычислений.

Чтобы избежать ошибок при программировании циклов, рекомендуется:

проверить, всем ли переменным, встречающимся в правой части операторов присваивания в теле цикла, присвоены до этого начальные значения, а также возможно ли выполнение других операторов;

проверить, изменяется ли в цикле хотя бы одна переменная, входящая в условие выхода из цикла;

предусмотреть аварийный выход из цикла по достижении некоторого количества итераций;

не забывать заключать в фигурные скобки тело цикла, если в нем требуется выполнить более одного оператора.

Операторы цикла в языке C++ взаимозаменяемы, но можно привести некоторые рекомендации по выбору наилучшего в каждом конкретном случае.

4. Наиболее важные моменты, которые следует запомнить

1. Выражение, стоящее в круглых скобках операторов `if`, `while` и `do while`, вычисляется в соответствии с приоритетами операций и преобразуется к типу `bool`.

2. Если в какой-либо ветви вычислений условного оператора или в цикле требуется выполнить более одного оператора, то они объединяются в блок.

3. Проверка вещественных величин на равенство опасна.

4. Чтобы получить максимальную читаемость и простоту структуры программы, надо правильно выбирать способ реализации ветвлений (с помощью `if`, `switch` или условной операции), а также наиболее подходящий оператор цикла.

5. Выражение, стоящее в скобках после ключевого слова `switch`, и константные выражения в `case` должны быть целочисленного типа.

6. Рекомендуется всегда описывать в операторе `switch` ветвь `default`.

7. После каждой ветви для передачи управления на конец оператора `switch` используется оператор `break`.

8. При написании любого цикла надо иметь в виду, что в нем всегда явно или неявно присутствуют четыре элемента: начальные установки, тело цикла, модификация параметра цикла и проверка условия продолжения цикла.

9. Если количество повторений цикла заранее не известно, необходимо предусматривать аварийный выход из цикла по достижении некоторого достаточно большого количества итераций.

5. Задания

Вариант 1

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + b & при\ \ x < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ \ x > 0\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) И (Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 2

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} \dfrac{1}{ax} - b & при\ \ x + 5 < 0\ и\ c = 0 \\ \dfrac{x-a}{x} & при\ \ x + 5 > 0\ и\ c \neq 0 \\ \dfrac{10x}{c-4} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац И Вц) ИЛИ (Вц И Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 3

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + bx + c & при\ \ a < 0\ и\ c \neq 0 \\ \dfrac{-a}{x-c} & при\ \ a > 0\ и\ c = 0 \\ a(x+c) & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

Ац И (Вц ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 4

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} -ax - c & \text{при } c < 0 \text{ и } x \neq 0 \\ \dfrac{x - a}{-c} & \text{при } c > 0 \text{ и } x = 0 \\ \dfrac{bx}{c - a} & \text{в остальных случаях} \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

Ац ИЛИ Вц ИЛИ Сц

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операция ИЛИ – поразрядная. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 5

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} a - \dfrac{x}{10 + b} & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x - a}{x - c} & \text{при } x > 0 \text{ и } b = 0 \\ 3x + \dfrac{2}{c} & \text{в остальных случаях} \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) И Сц

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 6

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + b^2 x & \text{при } c < 0 \text{ и } b \neq 0 \\ \dfrac{x + a}{x + c} & \text{при} > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац И Вц) ИЛИ (Ац И Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 7

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} -ax^2 - b & при\ x < 5\ и\ c \neq 0 \\ \dfrac{x-a}{x} & при\ x > 5\ и\ c = 0 \\ \dfrac{-x}{c} & в\ остальных\ случаях \end{cases}$$

где a, b, c – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) МОД2 (Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И, ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 8

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX

$$F = \begin{cases} -ax^2 & при\ \ c < 0\ и\ a \neq 0 \\ \dfrac{a-x}{cx} & при\ c > 0\ и\ a = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где a, b, c — действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац МОД2 Вц) И НЕ(Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И, ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 9

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + b^2 x & при\ a < 0\ и\ x \neq 0 \\ x - \dfrac{a}{x - c} & при\ a > 0\ и\ x = 0 \\ 1 + \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

НЕ(Ац ИЛИ Вц) И (Вц ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ, И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 10

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 - bx + c & при\ x < 3\ и\ b \neq 0 \\ \dfrac{x - a}{x - c} & при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

НЕ(Ац ИЛИ Вц) И (Ац МОД2 Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ, И, ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 11

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + \dfrac{b}{c} & при\ x < 1\ и\ c \neq 0 \\ \dfrac{x - a}{(x - c)^2} & при\ x > 1.5\ и\ c = 0 \\ \dfrac{x^2}{c^2} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац И Вц) МОД2 Сц

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 12

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^3 + b^2 + c & при\ x < 0.6\ и\ b + c \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0.6\ и\ b + c = 0 \\ \dfrac{x}{c} + \dfrac{x}{a} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) И Сц

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 13

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 + b & при\ x - 1 < 0\ и\ b - x \neq 0 \\ \dfrac{x-a}{x} & при\ x - 1 > 0\ и\ b + x = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) МОД2 (Вц И Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции И, ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 14

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} -ax^3 - b & при\ x + c < 0\ и\ a \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + c > 0\ и\ a = 0 \\ \dfrac{x}{c} + \dfrac{c}{x} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац МОД2 Вц) ИЛИ (Ац МОД2 Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 15

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} -ax^2 + b & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x}{x-c} + 5.5 & при\ x > 0\ и\ b = 0 \\ \dfrac{x}{-c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

НЕ(Ац ИЛИ Вц ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 16

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} a(x+c)^2 - b & при\ x = 0\ и\ b \neq 0 \\ \dfrac{x-a}{-c} & при\ x = 0\ и\ b = 0 \\ a + \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* — действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац МОД2 Вц) И НЕ(Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ, И, ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 17

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^2 - cx + b & при\ x + 10 < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + 10 > 0\ и\ b = 0 \\ \dfrac{-x}{a-c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац ИЛИ Вц) И НЕ(Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ, И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 18

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} ax^3 + bx^2 & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0\ и\ b = 0 \\ \dfrac{x+5}{c(x-10)} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

НЕ(Ац И Вц И Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ и И – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 19

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} a(x+7)^2 - b & при\ x < 5\ и\ b \neq 0 \\ \dfrac{x-cd}{ax} & при\ x > 5\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

где *a*, *b*, *c*, *d* – действительные числа.

Функция F должна принимать действительное значение, если выражение

(Ац МОД2 Вц) ИЛИ (Ац МОД2 Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции ИЛИ и МОД2 (сложение по модулю 2) – поразрядные. Значения *a*, *b*, *c*, *d*, Хнач., Хкон., dX ввести с клавиатуры.

Вариант 20

Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dX.

$$F = \begin{cases} -\dfrac{2x-c}{cx-a} & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ -\dfrac{x}{c} + \dfrac{-c}{2x} & \text{в остальных случаях} \end{cases}$$

где *a*, *b*, *c* – действительные числа.

Функция F должна принимать действительное значение, если выражение

НЕ(Ац ИЛИ Вц) И НЕ(Ац ИЛИ Сц)

не равно нулю, и целое значение в противном случае. Через Ац, Вц и Сц обозначены целые части значений *a*, *b*, *c*, операции НЕ, И и ИЛИ – поразрядные. Значения *a*, *b*, *c*, Хнач., Хкон., dX ввести с клавиатуры.

Вычисление функции с помощью разложения в ряд

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $x_{нач}$ до $x_{кон}$ с шагом `dx` с точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

1. $$\ln\frac{x+1}{x-1} = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots\right) \quad |x| > 1 \; s$$

2. $$e^{-x} = \sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots \quad |x| < \infty$$

3. $$e^{-x} = \sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots \quad |x| < \infty$$

4. $$\ln(x+1) = \sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} - \ldots \quad -1 < x \le 1$$

5. $$\ln\frac{1+x}{1-x} = 2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots\right) \quad |x| < 1$$

6. $$\ln(1-x) = -\sum_{n=1}^{\infty}\frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^4}{4} + \ldots\right) \quad -1 \le x < 1$$

7. $$\operatorname{arcctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} - \ldots \quad |x| \le 1$$

8. $$\operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} \ldots \quad x > 1$$

9. $$\operatorname{arctg} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \quad |x| \le 1$$

10. $$\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots \quad |x| < 1$$

11. $$\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots \quad |x| > 1$$

12. $$\operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \ldots \quad x < -1$$

13. $$\mathrm{e}^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \ldots \quad |x| < \infty$$

14. $$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \quad |x| < \infty$$

15. $$\frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} - \ldots \quad |x| < \infty$$

16. $$\ln x = 2\sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}} = 2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \ldots \right) \quad x > 0$$

17. $$\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \ldots) \;\; 0 < x \le 2$$

18. $$\ln x = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1)(x+1)^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2x^2} + \frac{(x-1)^3}{3x^3} + \ldots) \;\; x > \frac{1}{2}$$

19. $\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \ldots |x| < 1$

20. $\arccos x = \frac{\pi}{2} - \left(x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)} \right) =$

$$= \frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 8} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \ldots \right) \quad |x| < 1$$

Приложение

Основные операции языка C++

Операция	Краткое описание
Унарные операции	
`++`	увеличение на 1
`--`	уменьшение на 1
`sizeof`	размер
`~`	поразрядное отрицание
`!`	логическое отрицание
`-`	арифметическое отрицание (унарный минус)
`+`	унарный плюс
`&`	взятие адреса
`*`	разадресация
`new`	выделение памяти
`delete`	освобождение памяти
`(type)`	преобразование типа
Бинарные и тренарная операции	
`*`	умножение
`/`	деление
`%`	остаток от деления
`+`	сложение
`-`	вычитание
`<<`	сдвиг влево
`>>`	сдвиг вправо
`<`	меньше
`<=`	меньше или равно
`>`	больше
`>=`	больше или равно
`==`	равно
`!=`	не равно
`&`	поразрядная конъюнкция (И)
`^`	поразрядное исключающее ИЛИ
`|`	поразрядная дизъюнкция (ИЛИ)
`&&`	логическое И
`||`	логическое ИЛИ
`?:`	условная операция (тернарная)
`=`	присваивание
`*=`	умножение с присваиванием
`/=`	деление с присваиванием
`%=`	остаток отделения с присваиванием
`+=`	сложение с присваиванием
`-=`	вычитание с присваиванием
`<<=`	сдвиг влево с присваиванием
`>>=`	сдвиг вправо с присваиванием
`&=`	поразрядное И с присваиванием
`|=`	поразрядное ИЛИ с присваиванием
`^=`	поразрядное исключающее ИЛИ с присваиванием
`,`	последовательное вычисление