Практическое задание 5
Циклические алгоритмы и операторы цикла в C++

Цель работы – изучение синтаксических правил записи операторов цикла `do, while, for` и их семантики. Проектирование простых циклических алгоритмов и решение практических задач.

ПЛАН

1. Краткое теоретическое введение

Циклические алгоритмы

Алгоритмы порождаются прикладной задачей. В жизни процессы, которые являются циклическими, встречаются очень часто. Например, мы решаем задачи по математике, пока не выполним задание, маршрутное такси делает в день 7 поездок, профессор читает лекции раз в неделю. Общая идея – что-то сделать много раз.

Цикл, это процесс, который многократно повторяется, но обязательно:

- каждый раз с другими данными,
- и когда-то завершается.

Маршрутка каждый раз везет новых пассажиров, а после семи поездок водитель отдыхает. Задачи, как правило, решаются либо строго по заданию, либо когда ученик тренируется решать задачи.

Различают два вида циклических алгоритмов.

1. Арифметический цикл (управляемый счетчиком), как правило, повторяется заранее известное число раз.

Пример: сделан вклад в банке на 10 лет. Ежегодно вклад увеличивается, и так 10 раз.

Пример: найти сумму 15-ти слагаемых. Ровно 15 раз к сумме прибавляется очередное слагаемое.

2. Итерационный цикл (управляемый событием), как правило, заранее число повторений неизвестно.

Пример: вклад в банке сделан с целью накопления некоторой суммы. Ежегодно вклад увеличивается до момента накопления, но сколько пройдет лет, прежде чем это произойдет – заранее неизвестно.

Пример: найти сумму с указанной точностью. Многократно к сумме прибавляется очередное слагаемое, но сколько их – заранее неизвестно.

Управляющая переменная. В любом случае, чтобы цикл выполнялся правильно, нужно управление циклом. Управление выполняет некая величина, которая называется параметром цикла или управляющей переменной. Это одна из переменных программы, которая, как правило, имеет начальное значение, изменяется в теле цикла, определяет число повторений цикла и позволяет завершить его работу.

Логика выбора управляющей переменной полностью зависит от прикладной задачи. Научиться этому совсем не просто, но когда есть модель задачи, то для ее реализации есть операторы цикла.

Операторы цикла

Операторы цикла предназначены всего лишь для записи алгоритма. Поэтому прежде, чем кодировать, нужно описать процесс алгоритмически: найти управляющую переменную, определить ее начальное значение, закон изменения и условие завершения цикла.

Для описания арифметического цикла, чаще всего, используется оператор `for`, его синтаксис:

```
// В заголовке цикла – все управление.
for (Выражение1; Выражение2; Выражение3)
   {
   // Тело цикла;
   }
```

`Выражение1` задает начальное значение параметра цикла.

`Выражение2` (логическое) задает условие завершения выполнения цикла.

`Выражение3` задает приращение параметра цикла.

Тело цикла – как правило, составной оператор {}.

В заголовке цикла описано все управление или показан закон изменения параметра.

Для описания итерационного цикла, чаще всего, используются операторы `do` или `while`, синтаксис которых сходен:

```
while (Логическое_выражение)
   {
      // Тело цикла;
   }
```

```
do
{
   // Тело цикла;
} while(Логическое_выражение);
```

Если в цикле `for` управление в заголовке, то здесь управление явно должно быть задано законом изменения управляющей переменной. При входе в цикл она получает начальное значение, изменяется в теле цикла, и участвует в проверке условия завершения цикла.

2. Комментарии о семантике операторов цикла

Особенности `while`: проверка условия происходит до выполнения тела цикла, поэтому для заведомо ложного выражения тело цикла не будет выполнено ни разу.

Особенности `do…while`: проверка условия происходит после выполнения тела цикла, поэтому, как бы ни было задано `Логическое_выражение`, оператор тела цикла выполнится хотя бы один раз. Использование удобно, когда условие не определено при входе (управление событием).

Особенности `for`

1. Проверка условия происходит до выполнения тела цикла (как `while`).
2. Приращение управляющей переменной происходит после выполнения тела цикла.
3. Если условий выхода несколько, то выход происходит по первому условию.
4. Управляющая переменная не обязательно целого типа.

3. Практические задания

Создайте проект и выполните в нем последовательно три приведенные ниже упражнения.

<u>Упражнение 1</u>. Простой арифметический цикл

Запишите код программы для построения таблицы значений функции.

$$\mu_A(x) = \begin{cases} 1, & \text{если } Возраст(x) <= 20 \\ \dfrac{30 - Возраст(x)}{10}, & \text{если } 20 < Возраст(x) >= 30 \\ 0, & \text{если } Возраст(x) > 30 \end{cases},$$

где $Возраст(x) \in [15,35]$, $\Delta Возраст = 1$.

Формализация задачи

Определяется значение функции принадлежности для нечеткого утверждения.

Дано: диапазон изменения возраста персонажей. Сами персонажи (множество *x*) в этой задаче несущественны.

Найти: значения функции $\mu_A(x)$ в указанном диапазоне.

Управляющая переменная, это значение возраста. Закон изменения:
Возраст = [15,35], Δ=1., следовательно, это арифметический цикл.

В теле цикла вычисляется очередное значение функции и выводится на экран. Управление простое:

```
for (Age = 15; Age <=35; Age+=1)
   // Вычисление и вывод.
```

Упражнение 2.2. Простой итерационный цикл

Составить программу для определения наибольшего общего делителя (НОД) двух простых чисел *m* и *n* по алгоритму Евклида:

НОД = *m*, если *m* = *n*. Это признак завершения цикла. Иначе, если *m* > *n*, то *m* = *m* – *n*, иначе, если *m* < *n*, то *n* = *n* – *m*.

Значения n, m вводить в диалоге.

Упражнение 2.3. Цикл, управляемый событием

Предыдущая программа, это программа одного запуска: один запуск, один ввод и одно решение. Чтобы алгоритм можно было выполнить многократно, используйте цикл, управляемый событием. Событие, это выбор одного из возможных условий продолжения (завершения) цикла. Обычно, диалог с пользователем завершается выбором одного из двух действий: `OK` – принять и завершить, или `Cancel` – отказаться и завершить. Выход сопоставлен также нажатию клавиш `Enter` или `Esc`. Код нажатой клавиши можно распознать, в зависимости от его значения продолжить выполнение цикла или выйти: код `Enter` равен 13, код `Esc=` 27.

Сделайте код телом оператора `do` с выходом по условию

```
      cout << "ESC - выход\n");
} while (_getch() != 27);
```

Здесь `_getch()` – функция библиотеки `conio.h`, которая читает символ из входного потока и распознает его.

Кроме того, при вводе входного балла, чтобы отсечь ошибки ввода, обычно используется цикл для проверки на принадлежность диапазону. Например, для ввода числа в диапазоне от 1 до 5-ти, проверка может быть такой:

```
do
{
   cout << ”Введите балл =[1..5]\n”;
   cin >> Ball;
} while (Ball>=1 && Ball<=5);
```

В задании 3 проверка ввода может заключаться в проверке равенства *n=m,* что не имеет смысла.

Бонус

Напечатать в возрастающем порядке все числа в диапазоне от N_1 до N_2, в десятичной записи которых нет одинаковых цифр. Иметь возможность повторного обращения в диалоге.

Вопросы для самоконтроля

1. Что такое константа?
2. Почему в программе не описывается тип константы?
3. Каков синтаксис именованной константы?
4. Как узнать символьную константу?
5. Что такое ESC-последовательность?

6. Что такое переменная? Почему нужно объявлять все переменные?
7. Что такое тип данного?
8. Почему тип данного важен?
9. Какова роль объявления типа?
10. В каком месте кода нужно объявить переменную?
11. Что такое базовый тип, чем отличается базовый тип от производного типа?
12. Перечислите базовые типы данных.
13. Что такое модификатор типа?
14. Каков механизм модификатора long?
15. Каков механизм модификатора unsigned?
16. Что такое операция?
17. Что означает «операция возвращает значение»?
18. От чего зависит тип возвращаемого операцией значения?
19. Как можно классифицировать операции?
20. Что такое отношение, какого типа значение возвращают отношения?
21. Какие бывают логические операции? Какой тип должны иметь операнды логической операции?
22. Какие особенности имеют логические операции?
23. Какова особенность операции присваивания?
24. Сколько операндов имеет условная операция, какого типа значение она возвращает?
25. Что такое выражение?
26. Чем определяется тип выражения?
27. Чем определяется порядок вычисления выражений?
28. Что такое приведение типа?
29. Что такое преобразование типа?
30. Относится ли C++ к языкам со строгой типизацией?