Разветвляющиеся и циклические алгоритмы. Операторы передачи управления

Ветвление

Ветвления (разветвляющиеся алгоритмы, условные алгоритмы) разделяют процесс вычислений на два направления. Синтаксис:

```
if (условие)
     действие_1;
[else
     действие_2;]
```

условие — логическое выражение, для которого проверяется, истинно оно или ложно. Если условие истинно, то выполняется действие_1. В противном случае выполняется действие_2.
[] означают необязательность конструкции. Если ветви else нет, то такое ветвление называется неполным.

Если внутри одной из ветвей необходимо выполнить несколько действий, то они заключаются в фигурные скобки {}.

Пример (см. задачу 1): https://replit.com/@IekatierinaLat1/Conditionals#main.cpp

Работа со сложными условиями

Операторы сравнения возвращают значения логического типа bool, то есть могут принимать одно из двух значений — true (истина) или false (ложь). Когда условные выражения достаточно простые (например, a == b или x >= 5), то все просто. Но в случае, когда нужно, например, проверить принадлежность x к интервалу [a, b], необходимо учитывать преобразование типов.

Общие принципы преобразования целых значений в логические и обратно:

- Если преобразовать логическое true к типу int, то получится 1, а преобразование false даст 0.
- При обратном преобразовании число 0 преобразуется в false, а любое ненулевое число в true.

Если в выражении участвуют переменные двух типов данных, то результат будет приведен к более мощному типу. Например, если вы делите переменную типа int на переменную типа float, то результат будет иметь тип float. С логическим типом bool то же самое. Если в выражении есть int и bool, результат будет иметь тип int.

Пример (см. задачу 2): https://replit.com/@IekatierinaLat1/Conditionals#main.cpp

Тернарная операция

Существует тернарная операция, которой можно заменять простые ветвления. Синтаксис

```
условие? выражение_1: выражение_2
```

В зависимости от истинности условия будет возвращено либо значение выражения_1, либо значение выражения_2. Поскольку тернарная операция возвращает значение, ее можно использовать вместо простых ветвлений, когда нужно присвоить результат переменной в зависимости от выполнения условия.

Пример. Вычислить значение функции в точке. Функция: Если x>=0, то y=x, иначе y=-x.
Реализацию см. в задаче 3: https://replit.com/@IekatierinaLat1/Conditionals#main.cpp

Оператор выбора

В C++ нет оператора для множественного ветвления, однако есть оператор выбора. Он позволяет выбрать один (или несколько) вариантов из перечисленных. Синтаксис:

```
switch (выражение){
     case значение_1: действие_1; break;
     case значение_2: действие_2; break;
     ...
     case значение_n: действие_n; break;
     [default: действие;]
}
```

выражение — формально может принимать любое значение, но очень часто вместо выражения записывают только одну предварительно вычисленную переменную.
case — нужен для описания действий, которые должны выполниться, когда выражение приняло конкретное описанное значение. Значение_к может быть целочисленного, символьного или строкового типа.
break; нужен, чтобы не выполнялись лишние действия. Если его не записать, будут выполнены все действия, идущие за сработавшим case'ом.
default — [необязательное] действие по умолчанию. Выполняется, если не выполнился ни один из case'ов.
Пример (см. задачу 4): https://replit.com/@IekatierinaLat1/Conditionals#main.cpp

Цикл со счетчиком

Циклы со счетчиком предназначены для выполнения циклических действий, когда количество повторений цикла известно заранее или оно может быть вычислено из входных данных. Синтаксис:

```
for (инициализация; условие; модификация)
     действие;
```

Инициализация — присвоение начального значения счетчику цикла. **Счетчик цикла** — специальная переменная, значение которой будет изменяться перед началом каждой итерации (повторения) цикла.

Условие — условие продолжения цикла. Тело цикла выполняется, пока оно истинное.

Модификация — изменение счетчика цикла.
Примечания:

1. Строчка for (...) называется **заголовком** цикла, а все остальное — **телом** цикла.
2. Счетчиков в цикле может быть несколько, в таком случае они инициализируются через запятую в разделе инициализация, и модифицируются также через запятую в соответствующем разделе.
3. В C++ условие цикла может не зависеть от счетчика, поэтому, строго говоря, на самом деле цикл for это цикл с предусловием, но для удобства его называют циклом со счетчиком.
4. Счетчик цикла в C++ может быть нецелой переменной, что расширяет область применения цикла for.

Пример (см. задачу 1): https://replit.com/@IekatierinaLat1/Loops#main.cpp

Цикл с предусловием

Циклы с предусловием выполняются, пока условие истинно. Синтаксис:

```
while (условие)
     действие;
```

Следует помнить, что цикл с предусловием может не выполниться ни разу, если условие изначально было ложно, и выполняться бесконечно, если условие всегда будет истинным.

Пример (см. задачу 2): https://replit.com/@IekatierinaLat1/Loops#main.cpp

Цикл с постусловием

Цикл с постусловием также выполняется пока условие истинно, но он отличается от цикла с предусловием тем, что в нем сначала выполняется действие, а уже потом проверяется условие. Синтаксис:

```
do
     действие;
while (условие);
```

Примечания:

1. Цикл с постусловием всегда выполняется хотя бы один раз, поэтому его удобно использовать для решения задач, где обязательно выполнение какого-то набора действий.
2. Циклы с постусловием неудобно контролировать, т.к. условие находится после цикла. Поэтому рекомендуется использовать их только в тех случаях, когда тело цикла достаточно короткое. В остальных случаях следует предпочитать другие виды циклов.

Пример (см. задачу 3): https://replit.com/@IekatierinaLat1/Loops#main.cpp

Операторы передачи управления

Операторы передачи управления позволяют перемещаться в другую точку программы.
В C++ существуют четыре оператора передачи управления:

- Оператор безусловного перехода goto. Поскольку его использование нарушает принципы структурного программирования и затрудняет понимание кода, оператор строго не рекомендуется использовать в реальных проектах. Использование оператора безусловного перехода для решения учебных задач категорически запрещено!
- Оператор досрочного выхода break. Позволяет завершить выполнение цикла или оператора выбора. Внимание! На обычное ветвление он не действует! При использовании во вложенных циклах выход осуществляется только из самого внутреннего цикла, в котором написан break.
- Оператор перехода к следующей итерации цикла continue. Прекращает выполнение текущей итерации цикла и возвращается к его началу. Так же, как и break, работает только для одного цикла.
- Оператор возвращения значения return. Завершает выполнение функции и возвращает указанное значение. Более подробно будет рассмотрен в следующей лекции.

Пример: https://replit.com/@IekatierinaLat1/Transferring-control#main.cpp