Занятие 3.
Цикл while

Сегодня мы научимся писать код, который позволяет автоматически повторять одни и те же действия несколько раз. В программировании повторение действий называется циклом.

Цикл while

«While» переводится как «пока» и в программировании означает «пока условие верно, нужно выполнять следующие действия». В жизни цикл «пока» встречается очень часто. Например, вам срочно понадобились фиолетовые кроссовки для вечеринки. Как описать ваши действия для их получения? «Пока не найдёшь фиолетовые кроссовки – ходи по магазинам. Как только найдёшь – покупай». While очень похож на if, но если if выполняет действия один раз, то while будет делать их до тех пор, пока выполнено условие.

Рассмотрим этот цикл на примере. Есть число N. Необходимо вывести все числа по возрастанию от 1 до N. Как решить эту задачу? Нужно завести счётчик (переменную i), который будет равен текущему числу. Вначале это единица. Пока значение счетчика не превысит N, необходимо выводить его текущее значение и каждый раз увеличить его на единицу. В виде программы это записывается так:

```
int i, n;
cin >> n;
i = 1;
while (i <= n) {
    cout << i << " ";
    i = i + 1;
}
```

Когда выполнение программы доходит до закрывающей фигурной скобки, то происходит возврат к строке с while и проверка условия. Если оно верно, то блок команд выполняется ещё раз, а если нет, исполнение программы продолжается для команд, находящихся после блока команд while.

Рассмотрим ещё один пример. Например, компания друзей хочет купить пиццу. Чтобы не платить за доставку, нужно послать кого-нибудь в пиццерию. Кого? Самого младшего! Более формальная постановка задачи такая: среди последовательности чисел нужно найти и вывести минимальное. Последовательность состоит из положительных чисел и оканчивается нулём.
Как такую задачу будет решать человек? Каждый раз когда ему называют число, он сравнивает его с тем минимумом, который встречался раньше, и, если новое число ещё меньше, то запоминает его. Такие действия нужно продолжать пока числа в последовательности не закончатся. Вначале ему нужно запомнить самое первое названное число. В конце — назвать последнее, которое запомнил.

```
int min, now;
cin >> now;
min = now;
while (now != 0) {
    if (now < min) {
        min = now;
    }
    cin >> now;
}
cout << min;
```

Подсчёт суммы последовательности чисел

Решим ещё одну задачу: те же друзья, которые покупали пиццу, решили посчитать, хватит ли у них денег? Каждый говорит, сколько у него есть, и нужно посчитать, сколько их в сумме. Последовательность состоит из положительных чисел и оканчивается нулём, нужно вывести сумму всех элементов в последовательности.

Эта задача также очень легко решается человеком: достаточно помнить сумму уже названных чисел и каждое следующее число просто прибавлять к этой сумме. Сначала нужно запомнить ноль (числа ещё не заданы). Программа будет выглядеть так:

```
int sum = 0, now;
cin >> now;
while (now != 0) {
    sum = sum + now;
    cin >> now;
}
cout << sum;
```

Цифры числа

Задачи на подсчёт суммы, минимумов и максимумов встречаются в жизни очень часто. Например, задача по выявлению счастливого билетика. Если сумма цифр в первой половине номера совпадает с суммой цифр во второй половине, то билет считается счастливым (и его нужно съесть).

Научимся решать эту задачу по двум заданным половинам номера билета.
Мы легко можем научиться узнавать последнюю цифру числа — для этого достаточно посчитать остаток от его деления на 10. Также мы можем отбросить последнюю цифру числа — для этого нужно поделить число на 10 нацело. Таким образом мы сможем поочередно рассмотреть все цифры числа справа налево и остановиться тогда, когда цифры в числе закончатся (оно станет равным нулю). Сумму цифр можно посчитать так же, как и в предыдущей задаче. И сделать это нужно для каждой из половинок номера билета.

```
int part1, part2;
cin >> part1 >> part2;
int sum1 = 0, sum2 = 0;
while (part1 != 0) {
    sum1 = sum1 + part1 % 10;
    part1 = part1 / 10;
}
while (part2 != 0) {
    sum2 = sum2 + part2 % 10;
    part2 = part2 / 10;
}
if (sum1 == sum2) {
    cout << "Lucky ticket";
} else {
    cout << "unlucky :(";
}
```

Отладка программ

Мы дошли уже до достаточно сложных программ.

Если программа работает неправильно, то найти ошибку методом пристального взгляда иногда бывает непросто. Для облегчения поиска ошибок можно воспользоваться средствами отладки программ. На самом деле, на втором занятии мы уже пользовались такими средствами — когда устанавливали точку остановки (breakpoint) на команде return 0. Когда программа доходит до точки остановки, её выполнение прерывается и можно посмотреть текущие значения всех переменных.

Таким образом, если мы хотим понять, чему равны значения переменных в какой-то момент исполнения нашей программы, то можем установить breakpoint на той строке, где хотим прерваться. При запуске программы по F5 будут выполнены все команды до этой строки, а сама строка выделится красным. Если навести курсор на название любой переменной в программе, то будет показано её значение в момент до выполнения текущей строки. Чтобы продолжить выполнение программы нужно нажать F5 ещё раз — выполнение будет идти до очередного breakpoint.

Также можно выполнять программу пошагово, строку за строкой. Для этого нужно нажимать клавишу F10. При нажатии клавиши выполнится одна строка, выделенная в этот момент красным, и произойдёт переход к следующей команде.

Чаще всего отладка кода состоит в том, чтобы поставить breakpoint в начале потенциально ошибочного кода, запустить программу, проверить, что всё считалось правильно, а затем выполнять программу пошагово, смотреть на значения переменных и ловить момент, когда что-то пойдёт не так.

Вечный цикл

С помощью цикла while очень легко сделать вечный цикл: для этого достаточно написать условие, которое никогда не будет выполнено. Лет 30 назад, когда все работали в DOS, этого было достаточно, чтобы намертво повесить систему. Сейчас, к сожалению, можно просто нажать на крестик.

Например, можно взять задачу про вывод всех чисел от 1 до 100 и забыть об увеличении счётчика.

```
int i;
i = 1;
while (i <= 100) {
      cout << i << " ";
}
```

Эта программа будет работать вечно, потому что счётчик всегда будет равен единице. Чтобы прекратить работу программы можно нажать shift+F5 (одновременно) в окне Visual Studio. Лучше не закрывать чёрное окошко крестиком – так может что-нибудь сломаться.

Инструкции break и continue

Для управления поведением цикла можно использовать две инструкции, которые позволяют досрочно прерывать выполнение цикла или начинать выполнение инструкций цикла сначала.

Первая команда называется break. После её выполнения работа цикла прекращается (как будто не было выполнено условие цикла). Осмысленное использование конструкции break возможно, только если выполнено какое-то условие, то есть break должен вызываться только внутри if (находящегося внутри цикла). Использование break – плохой тон, по возможности, следует обходиться без него. Рассмотрим пример вечного цикла, выход из которого осуществляется с помощью break. Для этого решим задачу о выводе всех целых чисел от 1 до N. Использовать break таким образом ни в коем случае не нужно, это просто пример:

```
int n, i;
i = 1;
while (1) {
    cout << i;
    i = i + 1;
    if (i > 100)
        break;
}
```

Команда continue начинает исполнение тела цикла заново, начиная с проверки условия. Её нужно использовать, если начиная с какого-то места в теле цикла и при выполнении каких-то условий дальнейшие действия нежелательны. В задаче о покупке фиолетовых кроссовок использование continue может выглядеть так: мы заходим в очередной магазин в поисках фиолетовых кроссовок и продолжаем эти действия пока не найдём то, что нужно. Но если в магазине очередь в сто человек, то мы сразу же идём в следующий — независимо от того, есть ли в этом магазине кроссовки.

Приведём пример использования continue (хотя при решении этой задачи можно и нужно обходиться без него): дана последовательность чисел, оканчивающаяся нулём. Необходимо вывести все положительные числа из этой последовательности. Решение:

```
int now;
now = -1;
while (now != 0) {
    cin >> now;
    if (now <= 0)
        continue;
    cout << now << " ";
}
```

В этом решении есть интересный момент: перед циклом переменная инициализируется заведомо подходящим значением. Команда вывода будет выполняться только в том случае, если не выполнится условие в if.