Управляющие структуры в СИ

Управляющие структуры позволяют писать более сложные программы, нежели простое последовательное выполнение инструкций.

Всего в языке Си существует 7 видов управляющих структур, определяющих порядок выполнения операций:

- Последовательное выполнение
- Инструкция выбора if (…) { … }
- Инстуркция выбора if (…) { … } else { … }
- Выбор из набора значений switch (…) { case ... ; case ...}
- Цикл с предусловием while (…) { … }
- Цикл с постусловием do { … } while(…);
- Цикл со счётчиком for (… ; … ; …) { }

Ниже будут приведены примеры простых программ, демонстрирующих работу каждой из перечесленных управляющих структур.

Инструкция простого выбора if(…) { }

Предположим мы хотим написать компьютерную программу, которая давала бы нам словесное описание текущей температуры воздуха. Самый простой вариант будет выглядеть следующим образом:

```
#include <stdio.h>

int main()
{
    int temperature = 26;

    if(temperature > 25)
    {
        printf("It's hot.\n");
    }

    return 0;
}
```

Здесь мы видим объявление переменной temperature и инициализация её значением 26. После чего мы видим ключевое слово if и в скобках выражение для проверки (должно быть в скобках всегда). Здесь проверяется больше ли значение переменной temperature > 25, и если да, то выводиться сообщение о том, что в данный момент на улице жарко. В противном случае ничего не происходит.

Воспользовавшись знанием с прошлых лекций мы можем добавить интерактивности программе запросив значение температуры у пользователя:

```
#include <stdio.h>

int main()
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    if(temperature > 25)
        printf("It's hot.\n");

    return 0;
}
```

Замечание: если после инструкции if () всего лишь одна инстркция (в нашем случае это вызов функции printf(), то фигурные скобки можно не ставить.

Инструкция выбора if else

С увеличением интерактивности, информативность программы не увеличилась, если условие не выполняется то никакого вывода об этом нет. Изменим программу, добавив альтернативную ветку.

```
#include <stdio.h>

int main()
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    if(temperature > 25)
        printf("It's hot.\n");
    else
        printf("It cold!.\n");

    return 0;
}
```

Теперь если не выполняется условие, что температура больше 25 градусов, то будет выведено сообщение, что сейчас холодно.

Инструкции if else можно вкладывать друг в друга. Добавим в программу вывод, что если температура от 20 до 25 градусов, то вывести сообщение, что в данный момент тепло.

```
#include <stdio.h>

int main()
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    if(temperature > 25)
        printf("It's hot.\n");
```

```
    else
        if(temperature > 20 && temperature <= 25)
            printf("It's warm.\n");
        else
            printf("It cold!.\n");

    return 0;
}
```

Здесь во вложенной инструкции if мы можем увидеть применение операции И (&&). Таким образом сначала будет проверено, что температура больше 20 градусов, если это так, то будет проверено (если нет, то остальные условия не проверяются), что температура меньше или равно 25 градусам, и только в случае если это условие выполняется будет выведено сообщение о том, что в данный момент на улице тепло.

В противном случае будет выведено сообщение о том что на улице холодно.

Инструкция множественного выбора switch

С помощью вложенных друг в друга инструкций if ... else мы можем дать описание каждому конкретному значению температуру отдельное описание. Код программы будет иметь следующий вид:

```
#include <stdio.h>

int main()
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    if(temperature > 25)
        printf("It's hot.\n");
    else
        if(temperature == 25)
            printf("It's very warm.\n");
        else if(temperature == 24)
            printf("It's warm.\n");
        else if(temperature == 23)
            printf("It's cool.\n");
        else
            printf("It's cold.\n");

    return 0;
}
```

Примечание: Поскольку компилятор Си игнорирует пробельные символы (пробелы, табуляцию, перевод строки) то допустимо писать else и следующий вложенный if на одной строчке.

Чтобы упростить код в Си существует оператор и выбора из множества определенных значений. Тот же самый код при этом будет выглядеть следующим образом:

```
#include <stdio.h>

int main()
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    if(temperature > 25)
        printf("It's hot.\n");
    else
        switch(temperature)
        {
            case 25:
                {
                    printf("It's very warm.\n");
                    break;
                }
            case 24:
                {
                    printf("It's warm.\n");
                    break;
                }
            case 23:
                {
                    printf("It's cool.\n");
                    break;
                }
            default:
                {
                    printf("It's cold.\n");
                    break;
                }
        }

    return 0;
}
```

Важно, что break инструкции необходимы в каждом блоке case (в default необязательно). В противном случае будут выполнены инстркции всех нижестоящих блоков (впрочем иногда бывает полезно). Фигурные скобки ограничивающие инструкции в блоках case можно опустить, но если необходимо внутри блока объявлять переменные, то скобки обязательны.

Блок default отвечает за все остальные значения.

Цикл while

Предположим, нам необходимо написать программу, осуществляющую подсчет среднего из 5 введенных чисел. Одно из решений можно быть представлено с помощью использования инструкции повторения while

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;
```

```
    while(counter < 5)  //Пока значения счётчика введеных чисел меньше 5
    {
        printf("Enter number: ");
        scanf("%d", &num);          //Считать очередное число с клавиатуры

        sum = sum + num;            //Увеличить сумму на введенное число
        counter = counter + 1;      //Увеличить счётчик чисел.
    }

    printf("Average = %f\n", (double)sum / (double)counter);
    //Приведение целочисленных значений к вещественному типу.

    return 0;
}
```

Здесь мы объявили и инициализировали нулевыми значениями переменные для подсчёта чисел (counter), сохранения значения очередного введенного с клавиатуры значения (num), а также для сохранения суммы всех введенных чисел (sum).

Описания всех действий в каждой итерации цикла описано в коментариях (за //).

Часть выражений в коде можно сократить.

Так например выражение sum = sum + num; может быть сокращено до sum += num;

А counter = counter + 1 до counter++ или ++counter (что называется унарный инкремент, постфиксный в первом случае и префиксный во втором). Обновленный код будет выглядеть следующим образом.

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    while(counter < 5)
    {
        printf("Enter number: ");
        scanf("%d", &num);

        sum += num;
        counter++;
    }

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Цикл for

В тех случаях когда нам известно точное количество повторений цикла, удобнее использовать цикл со счётчиком for.

```
for (expression 1 ; expression 2; expression 3)
{
      statement 1;
      statement 2;
}
```

Здесь

- expression 1 — выражение для объявлений переменной или переменных счётчиков.
- expression 2 — условие продолжение цикла.
- expression 3 — операция увеличения (уменьшения) счётчика, выполняемая ПОСЛЕ всего тела цикла.
- statement 1 … N — тело цикла.

Для нашей задачи код будет следующим:

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    for( counter ; counter < 5; counter++) // for( ; counter < 5; counter++)
    {
        printf("Enter number: ");
        scanf("%d", &num);

        sum += num;
    }

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Прим: expression1 (объявление счетчика) в нашем случае можно оставить пустым (он уже объявлен и инициализирован), как указано закомментированной строчке.

Счётчиков в цикле for может быть больше одного и, начиная со стандарта C99 объявляться они могут прямо в самой инстркции for.

Пример использования двух счётчиков:

```
#include <stdio.h>

int main()
{
    for(int i = 0, j = 10; i != j; i++, j--)
        printf("i = %d, j = %d.\n", i, j);

   // printf("i = %d, j = %d.\n", i, j); ОШИБКА i и j не существуют вне цикла.

    return 0;
}
```

При попытке раскомментировать строчку после цикла for и скомпилировать программу мы получим ошибку, поскольку область видимости переменных i и j ограничена телом цикла.

Инструкции break и continue

Вернёмся к нашей задаче про среднее из чисел. Предположим, что мы хотим считать среднее только для положительных чисел, при этом не ограничивая их количество, а -1 использовать для завершения ввода. Для решения напрашивается следующий код:

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    while(num != -1)
    {
        printf("Enter number: ");
        scanf("%d", &num);

        sum += num;
        counter++;
    }

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Однако данный код не будет корректным поскольку когда мы введем -1, это значение будет добавлено к общей сумме и увеличен счетчик значений.

Есть несколько вариантов избежать подобной ситуации.

Первый вариант:

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    while(num != -1)
    {
        printf("Enter number: ");
        scanf("%d", &num);

        if(num < 0)  //Если введено отрицательное число вернуться к началу цикла
            continue;

        sum += num;
        counter++;
    }

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Инструкция continue прерывает выполнения текущей итерации и возвращет выполнение к проверке условия цикла.

Второй (менее красивый) вариант:

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    while(num != -1)
    {
        printf("Enter number: ");
        scanf("%d", &num);

        if(num == -1) //Если введено -1 выйти из цикла
            break;

        sum += num;
        counter++;
    }

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Инструкция break (выделена синим) передает управления первой инструкции после цикла. Выполнение цикла полностью прекращается.

Цикл do while

Иногда по логике алгоритма бывает понятно, что тело цикла должно быть выполнено хотя бы один раз. Как, например, в нашем предыдущем варианте решения задачи о среднем числе, хотя бы одно число должно быть введено. В таких случаях бывает удобно использовать цикл с пост условием do while. Для нашей задачи код может быть изменён так:

```
#include <stdio.h>

int main()
{
    int counter = 0;
    int sum = 0, num = 0;

    do
    {
        printf("Enter number: ");
        scanf("%d", &num);

        if(num >= 0)
        {
            sum += num;
            counter++;
        }
    }
    while(num != -1); //Точка с запятой обязательно!

    printf("Average = %f\n", (double)sum / (double)counter);

    return 0;
}
```

Функции

Функции в Си используются для структурирования программы и повторного использования одного и того же кода в разных частях программы. Чтобы продемонстрировать использование функций вернемся к нашей первой задаче про описание погоды.

Выделим в функцию отрывки кода, отвечающие за ввод данных и за печать информации на экран (в комментариях указаны номера, пояснение к которым приведено ниже).

```
#include <stdio.h>

int get_temperature() //1
{
    int temperature = 0;

    printf("Enter temperature: ");
    scanf("%d", &temperature);

    return temperature; //2
}

void print_temperature(int current_temperature) //3
{
    if(current_temperature > 25)
        printf("It's hot.\n");
    else
        switch(current_temperature)
        {
            case 25:
                {
                    printf("It's very warm.\n");
                    break;
                }
            case 24:
                {
                    printf("It's warm.\n");
                    break;
                }
            case 23:
                {
                    printf("It's cool.\n");
                    break;
                }
            default:
                {
                    printf("It's cold.\n");
                    break;
                }
        }

}


int main()
{
    int temperature = get_temperature(); //4
    print_temperature(temperature);      //5

    return 0;
}
```

1 — в данной строчке, int — тип возвращаемого значения, get_temperature — название функции. Пустые скобки обозначают, что у функции нет параметров.

2 — Поскольку функция возвращает целочисленное значение в её коде должна быть соотвтетсвющая инструкция return, которая прекращает выполнение функции и возвращает значение.

3 - void print_temperature(int current_temperature) — объявление функции печати информации о температуре. Т.к. функция ничего не возвращет, тип возвращаемого значения указывается как void. Функция принимает единственный параметр — значение текущей температуры, он указан в скобках.

4 — Объявление переменной temperature и определение (инициализация) её вызовом функции get_temperature(). Переменная temperature в функции main() и в функции get_temperature() - никак не связаны между сосбой и имеют разную область видимость.

5 — Вызов функции print_temperature с аргументом temperature — переменная, значение которой хранит введенную пользователем температуру.

В результате функция main() более наглядно отражает алгоритм работы программы (получить температуру, напечатать описание). Кроме того, если в программе где-то ещё понадобиться, например, ввод температуры, то достаточно будет вызвать функцию get_temperature(), не дублируя её код.