Лабораторная работа 1

Версия 0.8

Использование функции scanf для проверки успешности ввода переменных

В лабораторной работе 1 (и всех последующих) необходимо выполнять контроль данных, которые вводит пользователь. Этот контроль состоит из двух частей: во-первых, нужно убедиться, что введены данные нужного типа (например, что введено целое число), во-вторых, нужно проверить удовлетворяют ли эти данные условиям задачи (например, что число положительное).

Для решения первой задачи воспользуемся возможностями, которые предоставляет функция scanf.

Объявление функции scanf, согласно стандарту c99 (7.19.6.4), выглядит следующим образом:

```
#include <stdio.h>

int scanf(const char * restrict format, ...);
```

В качестве своего значения функция scanf возвращает количество переменных, которым было успешно присвоено значение. Это значение может быть меньше, чем количество указанных в вызове переменных (или даже равно нулю) из-за возникновения ошибки преобразования введенного значения к нужному типу. Если ввод значений окончился ошибкой еще до выполнения каких бы то ни было преобразований, функция возвращает значение EOF.

Рассмотрим несколько примеров

Пример 1. Файл input_1.c

```
#include <stdio.h>

int main(void)
{
    int rc, num;

    printf("Enter an integer: ");
    rc = scanf("%d", &num);
    if (rc == 1)
        printf("Your integer is %d\n", num);
    else
        printf("Input error\n");

    return 0;
}
```

Скомпилируем программу и запустим ее для разных входных данных.

Результаты работы input_1.exe

```
Enter an integer: 5
Your integer is 5

Enter an integer: a
Input error
```

```
ВНИМАНИЕ

Enter an integer: 100a
Your integer is 100
```

Вопросы для самопроверки:

1. Подумайте, что произойдет с символом “a”, который ввел пользователь?
2. Как этот символ обработает функция scanf?
3. Почему функция scanf не вернула код ошибки?

Пример 2. Файл input_2.c

```
#include <stdio.h>

int main(void)
{
    int rc, i_num;
    float f_num;

    printf("Enter an integer and a float: ");
    rc = scanf("%d %f", &i_num, &f_num);
    if (rc == 2)
        printf("Your integer is %d and your float is %f\n", i_num, f_num);
    else
        printf("Input error\n");

    return 0;
}
```

Скомпилируем программу и запустим ее для разных входных данных.

Результаты работы input_2.exe

```
Enter an integer and a float: 3 7.25
Your integer is 3 and your float is 7.250000

Enter an integer and a float: a 3
Input error

Enter an integer and a float: 3 a
Input error

ВНИМАНИЕ

Enter an integer and a float: 10 15.1d
Your integer is 10 and your float is 15.100000

Enter an integer and a float: 10.89
Your integer is 10 and your float is 0.890000
```

Вопросы для самопроверки

4. Объясните поведение функции scanf.

Как вы видите идеальное решение первой задачи только с помощью функции scanf невозможно, но полученное решение на лабораторных работах нас вполне устроит.

Для интересующихся (по материалам stackoverflow, вопрос 13049893).

Для проверки правильности ввода числа обычно рекомендуют использовать строковые функции, т.е. сначала вводится строка, а затем эта строка анализируется. Но можно попытаться обойтись только функцией scanf, усложнив алгоритм ее использования.

Пример 3. Файл input_3.c

```
#include <stdio.h>

int main(void)
{
    int rc, num;
    char tmp;

    printf("Enter an integer: ");

    // Считываем число и символ за ним (он обязательно будет)
    while(((rc = scanf("%d%c", &num, &tmp)) != 2 && rc != EOF) || tmp != '\n')
    {
        // Если мы здесь, то
        //  либо введено не число (не удалось прочитать число и символ);
        //  либо произошла ошибка в функции scanf (EOF);
        //  либо введено число, но сразу за ним указан какой-то символ.

        printf("Input error. Please, enter an integer: ");

        // Считываем символы до тех пор пока не найдем код клавиши Enter.
        // (это символы, которые не смогла обработать функция scanf).
        do
        {
            rc = scanf("%c", &tmp);
        }
        while(rc != EOF && tmp != '\n');
    }

    printf("Your integer is %d\n", num);

    return 0;
}
```

Скомпилируем программу и запусти ее для разных входных данных.

Результаты работы input_3.exe

```
Enter an integer: 5
Your integer is 5

Enter an integer: 125a
Input error. Please, enter an integer: 125
Your integer is 125

ВНИМАНИЕ (□ означает пробел)

Enter an integer: 15□
Input error. Please, enter an integer: 15
Your integer is 15
```

Вопросы для самопроверки

5. Понятен ли вам алгоритм, реализованный в третьей программе?
6. Какой недостаток есть у этого алгоритма? Как его устранить?