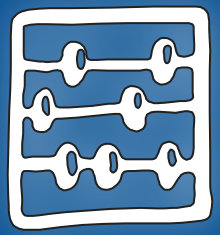
Алгоритмы и Алгоритмические Языки

Семинар #3:

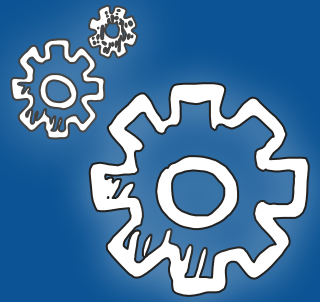
1. Результаты работы по системам счисления.
2. Типы языка Си, вывод данных в консоль.
3. Ввод данных в программу, проверка ввода.
4. Представление целых чисел в памяти компьютера.
5. Корректная программа, удобная программа.

17.09.2024

Результаты работы по системам счисления



Типы языка Си.
Вывод данных в консоль.



Типы данных языка Си

Целочисленные типы данных:

- Знаковые/беззнаковые
- Бывают разных размеров
- Размеры удовлетворяют неравенству

`long int`	at least **32**
`unsigned long int`	
`long long int` (C99)	at least **64**
`unsigned long long int` (C99)	

`char`	at least **8**
`signed char`	
`unsigned char`	
`short int`	at least **16**
`unsigned short int`	
`int`	at least **16**
`unsigned int`	

Besides the minimal bit counts, the C Standard guarantees that

`1` == `sizeof(char)` ≤ `sizeof(short)` ≤ `sizeof(int)` ≤ `sizeof(long)` ≤ `sizeof(long long)`.

Типы данных языка Си

Типы данных с плавающей точкой:

- **float –** одинарная точность (32 бита)
- **double** – двойная точность (64 бита)
- **long double** – расширенная точность (128 бит)

Строковый тип данных: **const char***

Булевый тип данных (значения – true/false): **bool**

Тип-пустышка: **void**

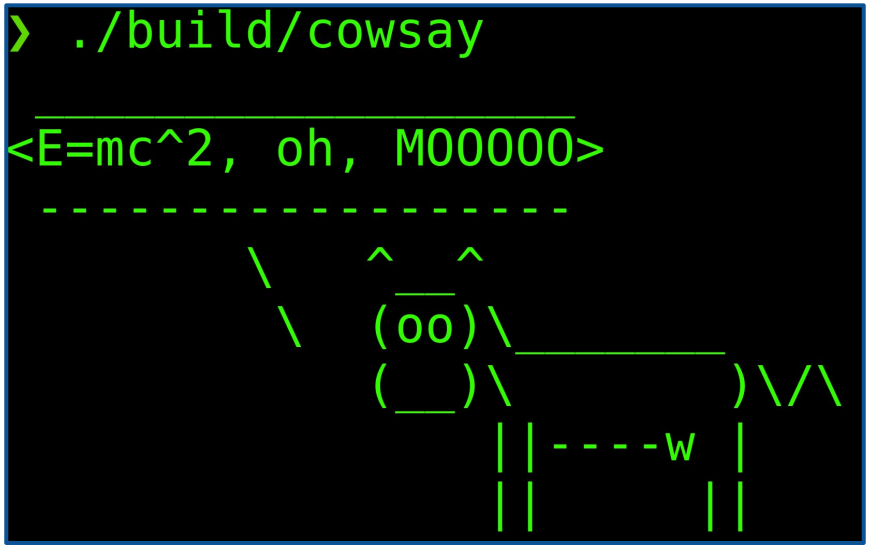
Вывод данных в консоль

putc – вывод единичного символа в консоль.

puts – вывод строки в консоль.

printf – форматированный вывод.

См. пример **03_cowsay**:



Ввод данных в программу. Проверка ввода.

Условие задачи в системе

Problem 00-2: A+B

На стандартном потоке ввода задаются два целых числа, не меньшие -32000 и не большие 32000.

На стандартный поток вывода напечатайте сумму этих чисел.

Числа задаются по одному в строке. Пробельные символы перед числом и после него отсутствуют. Пустые строки в вводе отсутствуют.

Examples

Input

```
1
2
```

Output

```
3
```

Простое решение задачи

```c
#include <stdio.h>

int main(void)
{
    // Declare variables:
    short num1, num2 = 0;

    // Parse program input:
    scanf("%hd\n%hd", &num1, &num2);

    // Print result:
    printf("%hd\n", num1 + num2);

    return 0;
}

```

Запуск простого решения

```
❯ cd path/to/examples/03_program_contracts
❯ gcc adder.c -o adder // Компиляция кода
❯ ./adder
1 // Жёлтым обозначен ввод данных от пользователя
2
3 // Зелёным – вывод программы
❯ ./adder
error
22940
```

При нарушении контракта программа выводит мусор!

Как пользователю понять, какой контракт был нарушен?

Проверка вводимых данных

Обратимся к документации на scanf. Можно посмотреть документацию, близкую к стандарту, на сайте **cppreference**:

C | File input/output

scanf, fscanf, sscanf, scanf_s, fscanf_s, sscanf_s

Defined in header <stdio.h>

```
int scanf( const char          *format, ... );                    (1)   (until C99)
int scanf( const char *restrict format, ... );                          (since C99)
```

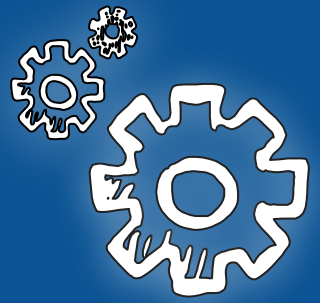
Return value

1-3) Number of receiving arguments successfully assigned (which may be zero in case a matching failure occurred before the first receiving argument was assigned), or EOF if input failure occurs before the first receiving argument was assigned.

Проверка вводимых данных

```c
#include <stdio.h>

int main(void)
{
    short num1, num2 = 0;

    // Parse program input:
    int num_inputs = scanf("%hd\n%hd", &num1, &num2);
    if (num_inputs < 2)
    {
        printf("adder: expected input \"<addentum1>\\n<addentum2>\\n\"\n");
        return 1;
    }

    // Print result:
    printf("%hd\n", num1 + num2);

    return 0;
}
```

Представление целых чисел в памяти компьютера.



Выход за границы представимости

Каково поведение программы при работе с большими числами?

```
❯ cd examples/03_program_contracts
❯ gcc adder.c -o adder
❯ ./adder
1
2
3
❯ ./adder
31500
31500
-2536
```

Представление целых чисел

Тип данных	Типичный размер	Представимые значения
`unsigned short`	16 бит	от 0 до 65535
`short`		от -32768 до 32767
`unsigned int`	32 бита	от 0 до $2^{32}-1$
`int`		от -2^{31} до $2^{31}-1$
`unsigned long`	64 бита	от 0 до $2^{64}-1$
`long`		от -2^{63} до $2^{63}-1$

Согласно стандарту, если в результате операции получается непредставимое значение, то фактический результат – зависит от представления чисел и не определён (Undefined Behavior).

Инверсный дополнительный код

Кодирование отрицательных чисел “знак+модуль”:

$value = (-1)^{sign} \times |value|$

$+1_{10} \leftrightarrow 00000001$ $\quad -0_{10} \leftrightarrow 10000000$ $\quad -127_{10} \leftrightarrow 11111111$

$+0_{10} \leftrightarrow 00000000$ $\quad -1_{10} \leftrightarrow 10000001$ $\quad +127_{10} \leftrightarrow 01111111$

Проблемы такого представления?

Инверсный дополнительный код:

Старший бит – знаковый.

$-x \leftrightarrow \sim x + 1$

$+0_{10} \leftrightarrow 00000000$ $\quad -0_{10} \leftrightarrow \sim 00000000 + 1 = 11111111 + 1 = 00000000$

$+9_{10} \leftrightarrow 00001001$ $\quad -9_{10} \leftrightarrow \sim 00001001 + 1 = 11110110 + 1 = 11110111$

Выход за границы представимости

```
31500 + 31500 = 0111_1011_0000_1100
              + 0111_1011_0000_1100
              = 1111_0110_0001_1000 = -536
```

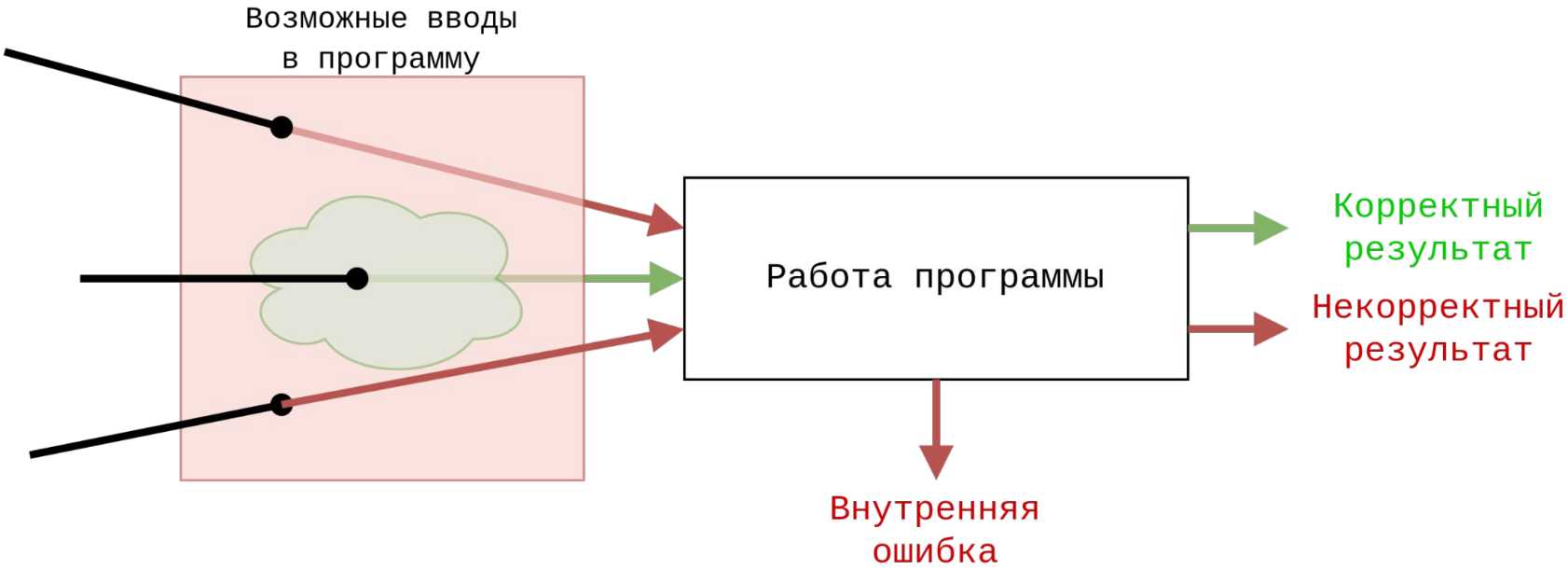
Проверка на переполнение:

```c
// Cast numbers to long and check if they overflow on addition:
int num1_ext = num1;
int num2_ext = num2;
int sum_ext = num1_ext + num2_ext;
if (sum_ext < SHRT_MIN || sum_ext > SHRT_MAX)
{
    printf("adder: %hd+%hd is too big to be represented in short", num1, num2);
    return 1;
}
```

Корректная программа, удобная программа.

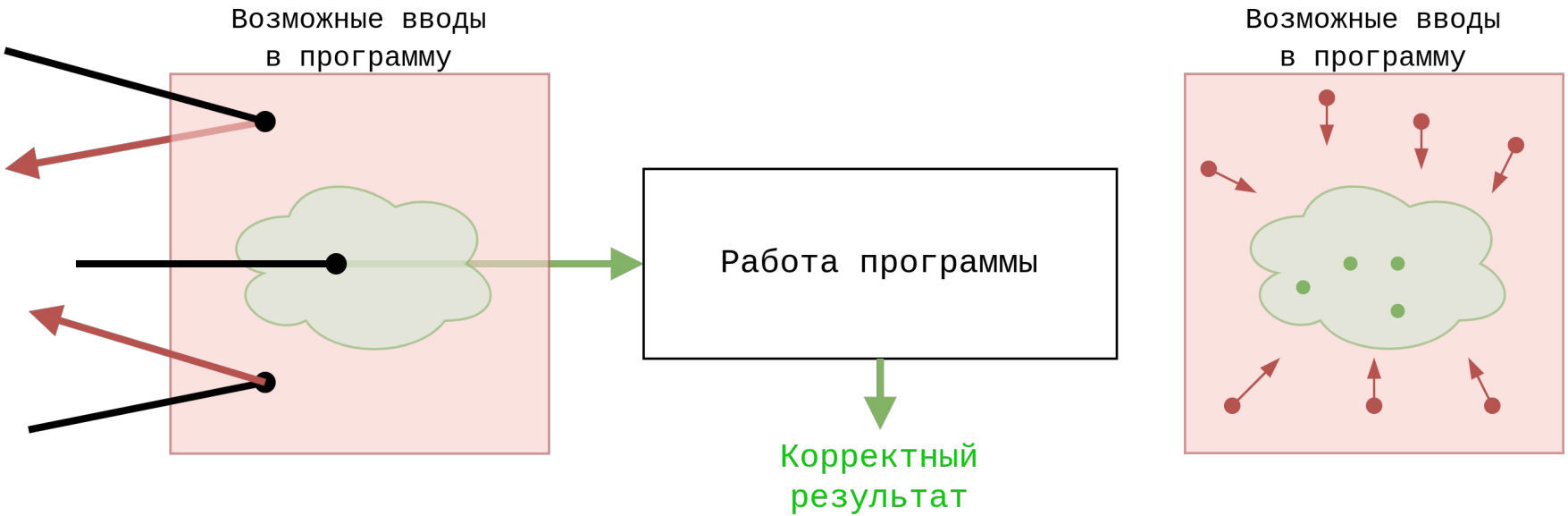
Корректная программа

Корректная программа – такая программа, которая для любого корректного ввода выдаёт вывод, соответствующий требованиям.

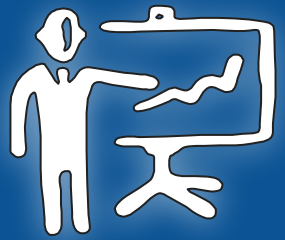


Удобная программа

Удобная программа – такая корректная программа, которая для любого некорректного ввода выдаёт понятное сообщение об ошибке.



Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com