Семинар 2+3: целые числа и ввод-вывод

19 Ноября, 2019

Беззнаковые числа

- Представляют из себя N-битные положительные целые числа на отрезке $[0, 2^N - 1]$
- Переполнение точно определено стандартом C (как сложение в $\mathbb{Z}_{2^N}$)
- $1111 + 0001 = 10000 = 0$

Как представлять отрицательные числа?

Signed magnitude representation

- Most significant bit (MSB) отвечает за знак: $0001 = +1$, $1001 = -1$
- Диапазон: $[-2^N + 1, 2^N - 1]$

Signed magnitude representation

- Most significant bit (MSB) отвечает за знак: $0001 = +1$, $1001 = -1$
- Диапазон: $[-2^N + 1, 2^N - 1]$
- Требуются отдельные операции сложения, вычитания и умножения для знаковых чисел
- Два представления для 0: $0000 = +0$ и $1000 = -0$

One's complement

- $-A = BitwiseNot(A)$
- Диапазон: $[-2^N + 1, 2^N - 1]$

One's complement

$$-1 = 1110$$
$$+2 = 0010$$

One's complement

$$-1 = 1110$$
$$+2 = 0010$$
$$1110 + 0010 = 10000 = 0$$

One's complement

$$-1 = 1110$$
$$+2 = 0010$$
$$1110 + 0010 = 10000 = 0$$
Упс...

One's complement: end-around-carry

$$1110 + 0010 = 10000 = 0 + 1 = 1$$

One's complement: недостатки

- Два представления для 0: $0000 = +0$ и $1111 = -0$.
- End-around-carry
- Зато сложение и вычитание одинаковое для знаковых и беззнаковых!

Two's complement

- Определение отрицательных чисел: $A + (-A) = 0$

Two's complement

- Определение отрицательных чисел: $A + (-A) = 0$
- Давайте каждому положительному числу сопоставим отрицательное

Two's complement

- Определение отрицательных чисел: $A + (-A) = 0$
- Давайте каждому положительному числу сопоставим отрицательное
- $-A = \mathit{BitwiseNot}(A) + 1$

Two's complement

- Определение отрицательных чисел: $A + (-A) = 0$
- Давайте каждому положительному числу сопоставим отрицательное
- $-A = \mathit{BitwiseNot}(A) + 1$
- Используется в современных процессорах
- Одно представление нуля:
  $-0 = \mathit{BitwiseNot}(A) + 1 = 1111 + 1 = 0000 = +0$
- Диапазон чуть больше: $[-2^N, 2^N - 1]$

Two's complement: недостатки

- Операции сравнения теперь сложные
- Умножение требует sign extension: 0010 = 00000010, 1000 = 11111000
- «Перекос» диапазона представимых чисел

Two's complement: недостатки

- Операции сравнения теперь сложные
- Умножение требует sign extension: 0010 = 00000010, 1000 = 11111000
- «Перекос» диапазона представимых чисел
- abs(INT_MIN) = ???

ZigZag representation

- LSB кодирует знак числа
- $(A << 1) \oplus (A >> 31)$
- Note that the second shift – the (A >> 31) part – is an arithmetic shift. So, in other words, the result of the shift is either a number that is all zero bits (if A is positive) or all one bits (if A is negative)
- Одно представление нуля
- $-1 = 1111 \oplus (1111 << 1) = 1111 \oplus 1110 = 0001$
- $1 = 0000 \oplus (0001 << 1) = 0000 \oplus 0010 = 0010$

ZigZag representation

- Вместе с varint-кодировкой экономит размер числа – не нужно хранить ведущие нули
- Используется в Google Protocol Buffers

Целые числа в C

Целые числа в C: типы данных

- Знаковые типы: short, int, long, ...
- Тоже знаковые типы: signed char, signed short, signed int, ...
- Беззнаковые типы: unsigned char, unsigned short, ...

Целые числа в C: типы фиксированных размеров

- Для каждого типа есть гарантии на диапазон, которое оно включает
- Но может включать больше!
- Типы фиксированного размера: int8_t, uint16_t, int32_t, uint64_t, ...
- Объявлены в inttypes.h
- Не работают, если платформа не поддерживает соотв. типы

Целые числа в C: немного мемов

- **char** читается как...

Целые числа в C: немного мемов

- **char** читается как... ['kar]

Целые числа в C: немного мемов

- **char** читается как... ['kar]
- **CHAR_BIT** — количество бит в байте

Целые числа в C: немного мемов

- **char** читается как... ['kar]
- **CHAR_BIT** — количество бит в байте
- Знаковость char...

Целые числа в C: немного мемов

- **char** читается как... ['kar]
- **CHAR_BIT** — количество бит в байте
- Знаковость char... **implementation defined**

Целые числа в C: немного мемов

- **char** читается как... ['kar]
- **CHAR_BIT** — количество бит в байте
- Знаковость char... **implementation defined**
- long вмещает числа из $[-2^{32}, 2^{32} - 1]$

Целые числа в C: про переполнение

- Переполнение signed-типов = **undefined behavior**!

Целые числа в C: про переполнение

- Переполнение signed-типов = **undefined behavior**!
- Иногда нужно проверить, было ли переполнение, но без UB

Целые числа в C: про переполнение

- Переполнение signed-типов = **undefined behavior**!
- Иногда нужно проверить, было ли переполнение, но без UB
- Семейство функций __builtin_add_overflow

```c
int a = ...;
int b = ...;
int res = 0;

if (__builtin_add_overflow(a, b, &res)) {
    // overflow occurs
} else {
    // res contains the result of a + b
}
```

Ввод-вывод в C

Текстовый поток — это последовательность символов, разбитая на строки, каждая из которых содержит ноль или более символов и завершается символом новой строки. Обязанность следить за тем, чтобы любой поток ввода-вывода отвечал этой модели, возложена на библиотеку: программист, пользуясь библиотекой, не должен заботиться о том, в каком виде строки представляются вне программы

Про признак окончания строки

- Разный под разными платформами
- \n — Unix
- \r — Mac
- \r\n — Windows

Про непотребства в терминале

- \r переводит каретку в начало строки, но не переносит саму строку
- Существуют специальные последовательности символов, которые изменяют состояние терминала
- Можно менять позицию курсора, цвет символов, цвет фона и даже издавать звук!

Порисуем в терминале!

Про escape sequences

Преимущества стандартной библиотеки C

- Функции стандартной библиотеки C легко вызвать из других языков (например, из ассемблера)
- Стандартный ввод-вывод C по умолчанию thread-safe
- Стандартный ввод-вывод C (иногда существенно) быстрее
- Форматный ввод-вывод C, несмотря на особенности, может оказаться удобнее в использовании

Вычеркните лишнее

Вычеркните лишнее

- Функции стандартной библиотеки C легко вызвать из других языков (например, из ассемблера)
- Стандартный ввод-вывод C по умолчанию thread-safe
- Стандартный ввод-вывод C (иногда существенно) быстрее
- Форматный ввод-вывод C, несмотря на особенности, может оказаться удобнее в использовании

Вычеркните лишнее

- Функции стандартной библиотеки C легко вызвать из других языков (например, из ассемблера)
- Стандартный ввод-вывод C по умолчанию thread-safe
- ~~Стандартный ввод-вывод C (иногда существенно) быстрее~~
- Форматный ввод-вывод C, несмотря на особенности, может оказаться удобнее в использовании

Если когда-то понадобится ускорить C++

- `ios_base::sync_with_stdio(false)`
- `cin.tie( nullptr )`

Thanks!