1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. «**Хранение типов данных**»
2. по дисциплине «Алгоритмизация и программирование»
3. Выполнил
4. студент гр. 5151003/30002 Штарев И. А.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Панков И. Д.

<*подпись*>

1. Санкт-Петербург
2. 2024

# Цель работы

Необходимо разработать программу, которая вычисляет размер, максимальное значение и минимальное значение основных типов данных языка Си. Разработанное программное обеспечение должно вычислять данные менее, чем за 1 секунду. Нельзя пользоваться функциями, которые позволяют определить размер типа данных (к примеру, sizeof(), константы, содержащиеся в библиотеке limits.h и пр.). Для выполнения условия по времени разрешено использовать встроенные функции замера времени.

# Постановка задачи

1. Разобраться, как хранятся целые числа в памяти компьютера (то есть, изучить прямой, обратный, дополнительный коды).
2. Разобраться, как хранятся числа с плавающей точкой в памяти компьютера (то есть, изучить представление таких чисел с помощью мантиссы и порядка).
3. Для типов данных int, short int, unsigned short int, char, unsigned char, unsigned int и unsigned long long int определить размеры в байтах.
4. Для типов данных int, short int, unsigned short int, char, unsigned char, unsigned int, unsigned long long int, float и double определить максимальное и минимальное значение.

# Теоретические исследования

## Хранение целых чисел

### Знаковый формат

В знаковом формате целые числа хранятся в двоичном виде, причем старший бит сегмента памяти, отведенного под то или иное число, зарезервирован для хранения знака. Знак «плюс» кодируется нулем, знак «минус» кодируется единицей. Пусть под хранение числа x в знаковом формате выделено n бит, тогда:

xmax = 2n–1 – 1

xmin = –2n–1

При использовании прямого кода число хранится вышеуказанным образом, без каких-либо изменений. Прямой код используется для хранения неотрицательных чисел. Для хранения отрицательных чисел используется дополнительный код. Перевод числа из прямого кода в дополнительный происходит в два этапа: перевод в обратный код (то есть, инвертирование всех битов, за исключением старшего) и прибавление единицы к младшему разряду. Для примера переведём число -17 (в однобайтовом формате) в дополнительный код:

* Прямой код: 100100012.
* Обратный код: 111011102.
* Дополнительный код: 111011112.

### Беззнаковый формат.

В беззнаковом формате старший бит не зарезервирован под хранение знака числа, поэтому в нём мы не можем хранить отрицательные числа, что компенсируется увеличенным в два раза (относительно знакового формата) диапазоном допустимых неотрицательных числа. Пусть под хранение числа x в беззнаковом формате выделено n бит, тогда:

xmax = 2n – 1

xmin = 0

## Хранение чисел с плавающей точкой

Для хранения чисел с плавающей точкой в форматах float и double выделяется 4 и 8 байт соответственно. Рассмотрим хранение в формате float. Старший бит, как и в случае с целочисленным знаковым форматом, выделяется под знак. Следующие 8 бит выделяются под хранение порядка числа, остальные 23 – под мантиссу. Для примера переведём число 4.13203 в тот вид, в котором он будет храниться на компьютере:

* Переведём целую часть в двоичный вид: 410 = 1002.
* Переведём дробную часть в двоичный вид: 0.1320310 = 0.001000011100110010110112. Заметим, что перевод этого числа можно продолжать до бесконечности, получая новые разряды бесконечной дроби. Мы имеем лишь 23 бита под хранение мантиссы, поэтому перевод был остановлен после получения первых 23 разрядов нужного двоичного числа. Важно отметить, что есть и числа, которые не превращаются в бесконечную дробь при переводе в двоичный вид (например, 0.2510 = 0.012).
* Переведём полученное двоичное число 100.001000011100110010110112 в экспоненциальную запись. Такая запись имеет вид (–1)S \* 1.M \* 10E, где S – знак степени, M – мантисса, E – порядок, причем все три записываются в двоичном виде. Сначала посчитаем порядок. Порядок – число сдвигов точки влево относительно изначальной позиции так, чтобы левее точки оказалась лишь одна единица. В нашем случае точку следует сдвинуть влево 2 раза, число будет иметь вид 1.00001000011100110010110112. Все, что осталось правее точки – мантисса числа, порядок равен 210 = 102. От мантиссы отсечем два младших бита, чтобы она поместилась в 23 бита. Итак,

S = 12

M = 000010000111001100101102

E = 102

* Если знак и мантисса записываются без изменений, то над порядком следует совершить дополнительные манипуляции. Учтём, что порядок может быть и отрицательным. Как было указано выше, целые отрицательные числа хранятся в дополнительном коде, и там это абсолютно оправданно, но в данном случае использование дополнительного кода создаст проблемы, связанные с тем, что процессору будет сложно определить, какое число больше, а какое меньше. Например, порядки -12 и 102 будут записаны как 111111112 и 000000102 соответственно; увидев старшие биты, процессор решит, что первый порядок больше второго, хотя на деле это не так. Иными словами, быстро (по старшим битам) сравнивать числа при записи порядка в дополнительном коде будет невозможно. Решением является хранение порядка относительно числа 12710. Таким образом, по модулю количество допустимых порядков сокращается вдвое, что компенсируется возможность хранить отрицательные порядки. Для получения порядка, записанного в таком виде, достаточно прибавить к нему 12710 и записать результат в прямом коде. В нашем случае: порядок 210 будет представлен как 210 + 12710 = 12910 = 100000012.
* Запишем полученные данные в 4-байтовый формат: 01000000100001000011100110010110.

Важно отметить следующие положения:

* Плюс и минус бесконечность записываются как числа, где в степени стоит 0 или 1 соответственно, порядок заполнен единицами, мантисса заполнена нулями.
* Если в процессе вычислений было получено несуществующее число (например, при делении бесконечности на бесконечность), оно будет записываться как число, в котором порядок заполнен единицами, мантисса – любое число отличное от нуля, знак значения не имеет. Такое число называется NaN (not a number).

## Определение максимальных и минимальных значений целочисленных типов данных

Рассмотрим два способа определить максимальные и минимальные значения целочисленных типов данных. Первый – создать переменную нужного типа, присвоить ей значение 0 и прибавлять по единице до тех пор, пока не произойдёт переполнение. Последнее число до переполнения – максимальное значение, первое после него – минимальное значение. Такая программа будет выполняться очень долго, что не подходит под заданное условием ограничение в одну секунду. Другой способ – вычислить искомые значения аналитически с помощью формул, описанных в пунктах 3.1.1 и 3.1.2. Такая программа будет работать очень быстро. Важно отметить, что в таком случае необходимо заранее знать, сколько памяти выделено под тот или иной тип данных.

# Тестирование и результаты работы программы

## Описание решения

Как было указано выше, для вычисления максимальных и минимальных значений требуется знать, сколько байт занимает переменная того или иного типа в памяти. Для получения этих данных воспользуемся указателями типа void. Они удобны тем, что им могут быть присвоены значения адресов переменных, тип которых заранее неизвестен. Итак, алгоритм решения поставленной задачи на языке Си заключается в следующем:

* Инициализировать переменную нужного типа.
* Инициализировать два указателя типа void.
* Присвоить первому указателю значения адреса инициализированной переменной.
* Присвоить второму указателю то же самое значение, только увеличенное на единицу.
* Инициализировать переменную типа long и присвоить ей значение разности второго указателя и первого указателя (используется тип long, так как язык Си считает результаты арифметических операций над указателями типом long). Полученное значение и будет размером переменной.
* Имея размер переменной, вычислить максимальное её значение с помощью вышеуказанных формул (в зависимости от того, знаковый или беззнаковый тип).
* Вычислить минимальное значение переменной, прибавив к максимальному единицу (произойдет переполнение переменной и значение сбросится к минимальному).
* Вывести полученные результаты в следующем формате:

int: byte, MAX\_INT, MIN\_INT

short int: byte, MAX\_INT, MIN\_INT

unsigned short int: byte, MAX\_INT, MIN\_INT

char: byte, MAX\_CHAR, MIN\_CHAR

unsigned char: byte, MAX\_UCHAR, MIN\_UCHAR

unsigned int: byte, MAX\_UINT, MIN\_UINT

unsigned long long int: byte, MAX\_LONG\_INT, MIN\_LONG\_INT

float: byte

double: byte

Важно отметить, что два предпоследних шага касаются лишь целочисленных типов данных, так как условие задачи не предполагает вычисление максимальных и минимальных значений для чисел с плавающей точкой.

Для вычисления времени работы программы была создана переменная типа clock\_t, хранящая значение времени перед исполнением всех вычислений. После исполнения вычислений была создана аналогичная переменная. В разности второй и первой хранится время, потраченное на вычисление. Для получения времени в секундах следует разделить полученное значение на макрос CLOCKS\_PER\_SEC.

## Тестирование и результаты работы программы

Результат выполнения программы:

A computer screen shot of numbers and digits

Description automatically generated

Рисунок 1. Вывод результатов программы

# Выводы

В ходе выполнения данной работы было изучено, как целые и дробные числа хранятся в памяти компьютера, как вычисляется количество байт, выделяемое под них в памяти, максимальные и минимальные значения целочисленных переменных. Была создана программа, вычисляющая названные значения наиболее оптимальным образом.

Список используемых источников

1. Брайан Керниган, Деннис Ритчи «Язык программирования Си», - http://www.r-5.org/files/books/computers/languages/c/kr/Brian\_Kernighan\_Dennis\_Ritchie-The\_C\_Programming\_Language-RU.pdf
2. clock | Microsoft Learn -

https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/clock?view=msvc-170

1. Alek OS «Как работают отрицательные числа | Основы программирования» -

https://youtu.be/BIYiuy8WWiU

1. Alek OS «Как работают числа с плавающей точкой | Основы программирования» -

https://youtu.be/U0U8Ddx4TgE

ПРИЛОЖЕНИЕ А

Листинг программы «**Хранение типов данных в языке Си**»

#include <stdio.h>

#include <time.h>

#include <math.h>

int main() {

clock\_t time = clock();

//int

int integer[2], n1, n2, IRes;

n1 = &integer[1], n2 = &integer[0];

IRes = n1 - n2;

IRes = pow(2, IRes \* 8 - 1);

printf("int: %d, %d, %d\n", n1 - n2, IRes - 1, IRes);

//short int

short int sinteger[2], SIRes;

n1 = &sinteger[1], n2 = &sinteger[0];

SIRes = n1 - n2;

SIRes = pow(2, SIRes \* 8 - 1);

printf("short int: %d, %hi, %hi\n", n1 - n2, SIRes - 1, SIRes);

//unsigned short int

unsigned short int usinteger[2], USIMin = 0;

n1 = &usinteger[1], n2 = &usinteger[0];

printf("unsigned short int: %d, %hu, %hu\n", n1 - n2, USIMin-1, USIMin);

//char

char Char[2], CharRes, CharMax;

n1 = &Char[1], n2 = &Char[0];

CharRes = n1 - n2;

CharRes = pow(2, CharRes \* 8 - 1);

CharMax = CharRes - 1;

printf("char: %d, %d, %d\n", n1 - n2, CharMax, CharRes);

//unsigned char

unsigned char uchar[2], UCMin = 0;

n1 = &uchar[1], n2 = &uchar[0];

printf("unsigned char: %d, %hhu, %hhu\n", n1 - n2, UCMin-1, UCMin);

//unsigned int

unsigned int uinteger[2], UIMin = 0;

n1 = &uinteger[1], n2 = &uinteger[0];

printf("unsigned int: %d, %u, %u\n", n1 - n2, UIMin - 1, UIMin);

//unsigned long long int

unsigned long long int ullinteger[2], ULLIMin = 0;

n1 = &ullinteger[1], n2 = &ullinteger[0];

printf("unsigned long long int: %d, %llu, %llu\n", n1 - n2, ULLIMin - 1, ULLIMin);

//float %f

float Float[2];

n1 = &Float[1], n2 = &Float[0];

printf("float: %d\n", n1 - n2);

//double %f

double Double[2];

n1 = &Double[1], n2 = &Double[0];

printf("double: %d\n", n1 - n2);

time = clock() - time;

printf("\n%f", (double)time / CLOCKS\_PER\_SEC);

return 0;

}