Государственное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

090302 200008 006

**«Технологии обработки информации»**

Семестр 4

**ОТЧЁТ**

по лабораторным работам

Лабораторная работа №1



Преподаватель: Плотников В.Ю.

Студент : Киселев В.А.

Группа : ФТ — 200008

Екатеринбург 2022 г.

Оглавление

[Цель работы 3](#_Toc104577268)

[Постановка задачи 3](#_Toc104577269)

[Средства разработки 3](#_Toc104577270)

[Описание алгоритма 3](#_Toc104577271)

[Реализация 6](#_Toc104577272)

[С++ 6](#_Toc104577273)

[Python 6](#_Toc104577274)

[JavaScript 7](#_Toc104577275)

[Результаты 8](#_Toc104577276)

[C++ 8](#_Toc104577277)

[Python 8](#_Toc104577278)

[JavaScript 8](#_Toc104577279)

[Анализ результатов 8](#_Toc104577280)

[Выводы 8](#_Toc104577281)

# Цель работы

Экспериментально установить пределы представимых значений для целочисленных типов данных в следующих языках программирования: C++ (для типов char, short, int, long и long long), Python и JavaScript. На основании полученных результатов сделать заключение о способе представления и размере. Оформить отчёт о проделанной работе.

# Постановка задачи

1) Разработать алгоритм

2) Реализация алгоритма

3) Провести эксперимент

4) Анализ результатов

5) Выводы

# Средства разработки

1) CLion 2021.3.3

2) PyCharm Professional 2021.3.2

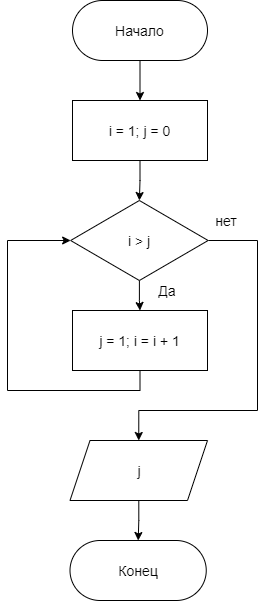
3) WebStorm 2021.3.2

4) Google Chrome 98.0.4758.102

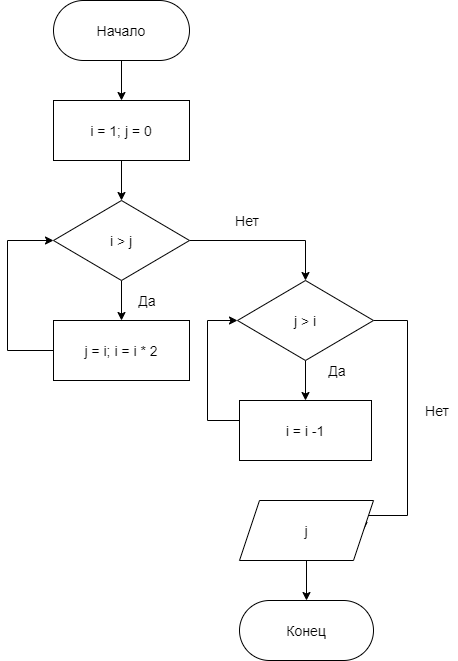
# Описание алгоритма

Модель экспоненциального поиска будет использоваться для разработки решения поставленной задачи. С самого начала нам нужно найти верхнюю границу char, short, int, long и long long. На первом этапе мы создаем цикл, в котором увеличиваем число до тех пор, пока оно является представимым. При этом прибавляемый к числу шаг увеличиваем каждый раз, когда получаем большее число. Когда число «вышло» за границу диапазона, уменьшаем шаг в два раза и начинаем новый цикл. Здесь к полученному в первом цикле числу прибавляем новый шаг. Если число увеличилось, уменьшаем шаг еще раз и опять прибавляем. В случае, если шаг получился меньше 1, делаем его равным 1. Если при добавлении шага мы не получаем большее число, то при шаге меньшим или равным единице завершаем цикл. Таким образом получаем искомое значение. Аналогичные действия совершаются для поиска нижней границы, только шаг при этом вычитается.

Для типов данных с малым диапазоном (char, short, int) можно смело предположить, основываясь на их названиях, что достаточно воспользоваться бесконечным циклом, в котором последующие значение будет больше предыдущего.



Для типов с большим диапазоном (long, long long) воспользуемся экспоненциальным поиском – алгоритмом для поиска по бесконечным или заранее неопределенным массивам.



# Реализация

## С++

#include <iostream>  
#include <typeinfo>  
  
template <typename T>  
T search\_max\_value(T& value) {  
 T max\_value = 0;  
 while (value > max\_value) {  
 max\_value = value;  
 value \*= 2;  
  
 };  
 while (value < max\_value) value--;  
 return value;  
}  
  
template <typename T>  
T search\_min\_value(T& value) {  
 value \*= -1;  
 T min\_value = 0;  
 while (value < min\_value) {  
 min\_value = value;  
 value \*= 2;  
  
 };  
 value = min\_value + 1;  
 while (value > min\_value) value--;  
 return value;  
}  
  
void output(auto & var){  
 if ((std::string)typeid(var).name() == (std::string)"char"){  
 std::cout << typeid(var).name() << " min:" << (int)search\_min\_value(var) << ", max:" << (int)search\_max\_value(var) << std::endl;  
 } else {  
 std::cout << typeid(var).name() << " min:" << search\_min\_value(var) << ", max:" << search\_max\_value(var) << std::endl;  
 }  
}  
  
int main() {  
 char var1 = 1;  
 short var2 = 1;  
 int var3 = 1;  
 long var4 = 1;  
 long long var5 = 1;  
  
 output(var1);  
 output(var2);  
 output(var3);  
 output(var4);  
 output(var5);  
  
 return 0;  
}

## Python

# Maximum  
  
base = 1  
temp = 0  
step = 1  
while True:  
 temp = base + step  
 if temp > base:  
 base = temp  
 step = step \* step / 2 + 1  
 else:  
 break  
  
print(base)  
  
# Minimum  
  
base = -1  
temp = 0  
step = 1  
while True:  
 temp = base - step  
 if temp < base:  
 base = temp  
 step = step \* step / 2 + 1  
 else:  
 break  
  
print(base)

## JavaScript

let ***base*** = 1;  
let ***temp*** = 0;  
  
while (***base*** > ***temp***) {  
 ***temp*** = ***base***;  
 ***base*** = 2 \* ***base***;  
}  
***console***.log(***temp***);  
  
***base*** = -1;  
***temp*** = 0;  
while (***base*** < ***temp***) {  
 ***temp*** = ***base***;  
 ***base*** = 2 \* ***base***;  
}  
***console***.log(***temp***);

# Результаты

## C++

Изображение выглядит как текст

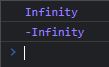
Автоматически созданное описание

## Python

Изображение выглядит как текст

Автоматически созданное описание

## JavaScript



# Анализ результатов

В процессе лабораторной работы мы установили размеры значений char, short, int, long, long long.

Char: диапазон значений [-128,127]. Множество содержит 256 элементов, следовательно, переменная типа char занимает в памяти 1 байт.

Short: диапазон значений [-32768,32767]. Множество содержит 65536, следовательно, переменная типа short занимает в памяти 2 байта.

Int: диапазон значений [-2147483648,2147483647]. Множество содержит 4294967296, следовательно, переменная типа int занимает в памяти 4 байта.

Long: диапазон значений [-2147483648,2147483647] Множество содержит 4294967296, следовательно, переменная типа long занимает в памяти 4 байта.

Long long: диапазон значений [9223372036854775808, 9223372036854775807] Множество содержит 18446744073709551616, следовательно, переменная типа long long занимает в памяти 8 байт.

# Выводы

В данной лабораторной работе я экспериментально установил пределы представимых значений для целочисленных типов данных на трех языках программирования.