

Федеральное государственное автономное образовательное учреждение
высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Лабораторная работа №1

по дисциплине «Теоретические основы информатики»

Кодирование информации и представление данных в памяти компьютера

Выполнил: студент Яцишин Л.С., уч. группа ПСАПР-25-2

Пермь, 2025

Содержание

1	Задание 1	3
	Задание 1 — трассировка	3
	Задание 1 — скриншоты	3
2	Задание 2	4
	Задание 2 — трассировка	4
	Задание 2 — скриншоты	4
3	Задание 3	6
	Задание 3 — трассировка	6
	Задание 3 — C++	7
	Задание 3 — Python	9
	Задание 3 — Проверка на других значениях N	10

1 Задание 1

Сколько раз выполнится цикл в программе, фрагмент кода которой на языке C++ приведён ниже, если переменная A имеет целочисленный тип – целое без знака в формате байта (контроль выхода за допустимый диапазон значений отключён). Какое значение примет переменная A после завершения цикла?

Поясните ответ – выполните трассировку программы («сухую прокрутку» – пошаговое выполнение вручную) – заполните таблицу, структура которой показана ниже, чтобы обосновать свой ответ (покажите, как меняется значение переменной – как выполняются операции – повторите их столько раз, сколько раз они повторятся при выполнении цикла):

```
1 static unsigned char A;  
2 A = 255;  
3  
4 do { A++;  
5 } while (A != 0);
```

Трассировка

Оператор/ операция	Десятичное значение переменной A: ожидаемое/полученное	Внутреннее представление A	Комментарий
A = 255;	255/255	11111111 ₂ 0xFF	Инициализация
A++	256/0	00000000 ₂ 0x00	Переполнение
A != 0	ложь	00000000 ₂ 0x00	Условие не выполнилось
Цикл не повторился			

Итого: тело цикла выполнено ровно 1 раз; после завершения A == 0.

Скриншоты

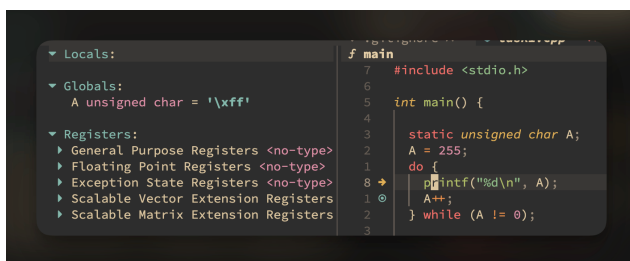


Рис. 1: static unsigned char A = 255;



Рис. 2: A++;

Выводы по заданию 1

- Трассировка подтверждается отладкой: после единственного прохода A становится 0.
- Причина поведения – определённое поведение беззнаковых типов в C/C++. Арифметика для unsigned типов определяется по модулю 2^N (здесь $N = 8$), поэтому $11111111_2 + 1_{10} = 0$
- Вывод: цикл выполняется один раз, финальное значение A == 0.

2 Задание 2

Задание 2. Сколько раз выполнится цикл в программе, фрагмент кода которой на языке C++ приведён ниже, если переменная A имеет целочисленный тип – целое со знаком в формате байта (контроль выхода за допустимый диапазон значений отключён).

Поясните ответ – выполните трассировку программы («сухую прокрутку») – заполните таблицу, чтобы обосновать свой ответ (покажите, как меняется значение переменной – повторите их столько раз, сколько раз они повторятся при выполнении цикла):

```
1 static signed char A;
2
3 A = -127;
4 while (A < 0) {
5     A = A - 1;
6 }
```

Трассировка

Оператор/ операция	Десятичное значение переменной A: ожидаемое/полученное	Внутреннее представление A	Комментарий
A = -127;	-127 / -127	10000001 ₂ 0x81	Инициализация
A < 0	истина	10000001 ₂ 0x81	Условие выпол- нилось
A = A - 1;	-128 / -128	10000000 ₂ 0x80	Тело цикла выполнилось
A < 0	истина	10000000 ₂ 0x80	Условие выпол- нилось
A = A - 1;	-129 / 127	11111111 ₂ 0x7F	Переполнение
A < 0	ложь	11111111 ₂ 0x7F	Условие не вы- полнилось

Скриншоты



Рис. 3: static signed char A = -127;

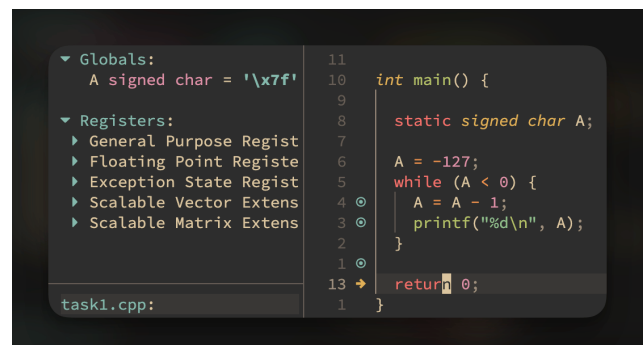


Рис. 4: A = -128;

Выводы по заданию 2

- Трассировка и отладка совпали: тело цикла выполняется дважды. Последовательность значений: $-127 \rightarrow -128 \rightarrow 127$, после чего условие $A < 0$ ложно и цикл прекращается.
- Причина: тип `signed char` имеет диапазон $[-128, 127]$. Первое вычитание даёт -128 . Второе вычитание для -128 выходит за диапазон знакового байта.
- Поэтому в запуске без проверок получается $10000000_2 - 00000001_2 = 01111111_2 = 127$ и завершение цикла на $127 > 0$. Итог: цикл выполнился два раза, финальное значение $A == 127$.

3 Задание 3

Сколько раз выполнится цикл в программе, фрагмент кода которой на языке Pascal приведён ниже, если переменная S имеет вещественный тип однократной точности, а переменная N – тип целого без знака в формате байта. Какое значение получит переменная S?

Поясните ответ – выполните трассировку программы (пошаговое выполнение вручную – «сухую прокрутку») – заполните таблицу, структура которой приведена ниже, вычисляя значения при выполнении каждого оператора, чтобы обосновать свой ответ (покажите, как меняется значение переменной – повторите их столько раз, сколько раз они повторяются при выполнении цикла, и заполните соответствующие строки в таблице)

```

1 {NOTE: real – двухкратная точность, single – однократная }
2 var S: single; N: byte; { Объявление переменных }
3 begin
4   N := 3; { Присваивание переменной N значения }
5   S := 1/N; { Присваивание переменной S значения – вычисляется выражение 1/N }
6   while S <> 1 do {Выполнять, пока S не равно 1 }
7     begin
8       writeln(S); {вывести на экран}
9       S := S + 1 / N;
10    end;
11    writeln(S); {вывести на экран}
12 end.
```

Трассировка

Оператор/ операция	Десятичное значение S: ожидаемое/полученное	Внутреннее представление S	Комментарий
N := 3;	—	—	Инициализация
S := 1/N;	0.33333334/0.33333334	0x3EAAAAAB	1/3 округлён до ближайшего representable
S <> 1	истина	0x3EAAAAAB	Условие цикла выполнено
S := S + 1/N;	0.6666667/0.6666667	0x3F2AAAAB	Сумма двух ап- проксимаций 1/3
S <> 1	истина	0x3F2AAAAB	Ещё не 1
S := S + 1/N;	1.0/1.0	0x3F800000	Округление даёт ровно 1.0
S <> 1	ложь	0x3F800000	Выход из цикла

Итого по трассировке. Тело цикла выполняется 2 раза. Последовательность значений:

$$S : \frac{1}{3} (\approx 0.33333334) \rightarrow \frac{2}{3} (\approx 0.6666667) \rightarrow 1.0,$$

после чего условие $S <> 1$ ложно.
(проверял в онлайн компиляторе [Online Pascal Compiler](#))

Перевод на C++ и сравнение

Переведите программу на язык C++, выбрав подходящие типы данных и операторы языка C++. Выполните «сухую прокрутку» программы на C++ (заполните трассировочную таблицу, заменив операторы языка Pascal на соответствующие операторы языка C++). Какие результаты получены? С какой точностью (сколько десятичных знаков) могут быть эти числа записаны в памяти компьютера в указанном формате? Проверьте, используя справочную систему Microsoft (информацию по типам данных в C++).

В чём разница в описаниях типов данных, чем различаются правила выполнения операций на языках C++ и Pascal? Объясните ответ. Проверьте результаты, выполнив программу на Pascal.

```
1  #include <stdio>
2
3  int main() {
4      float S;
5      unsigned char N = 3;
6
7      S = 1.0f / N;
8
9      int iters = 0;
10     while (S != 1.0f) {
11         printf("iters=%d, S=%.9E\n", iters, S);
12         S = S + 1.0f / N;
13         ++iters;
14     }
15
16     printf("iters=%d, S=%.9E\n", iters, S);
17 }
```

Оператор/ операция	Десятичное значение S: ожидаемое/полученное	Внутреннее представление S	Комментарий
unsigned char N = 3;	—	—	Инициализация
S = 1.0f / N;	0.33333334/0.33333334	0x3EAAAAAB	1/3 округлѐн до ближайшего representable
S != 1.0f	истина	0x3EAAAAAB	Условие цикла выполнено
S = S + 1.0f / N;	0.66666667/0.66666667	0x3F2AAAAAB	Сумма двух аппроксимаций 1/3
S != 1.0f	истина	0x3F2AAAAAB	Ещё не 1
S = S + 1.0f / N;	1.0/1.0	0x3F800000	Округление даёт ровно 1.0
S != 1.0f	ложь	0x3F800000	Выход из цикла

```

Target OS: Linux for x86-64
Compiling main.pas
Linking a.out
10 lines compiled, 0.0 sec
3.333333433E-01
6.666666865E-01
1.000000000E+00

```

Рис. 5: Выполнение на Pascal

```

toi/lab1 main* < runcpp -l -d task1.cpp
Learning mode (no optimizations, with
iters=0, S=3.333333433E-01
iters=1, S=6.666666865E-01
iters=2, S=1.000000000E+00
toi/lab1 main* >

```

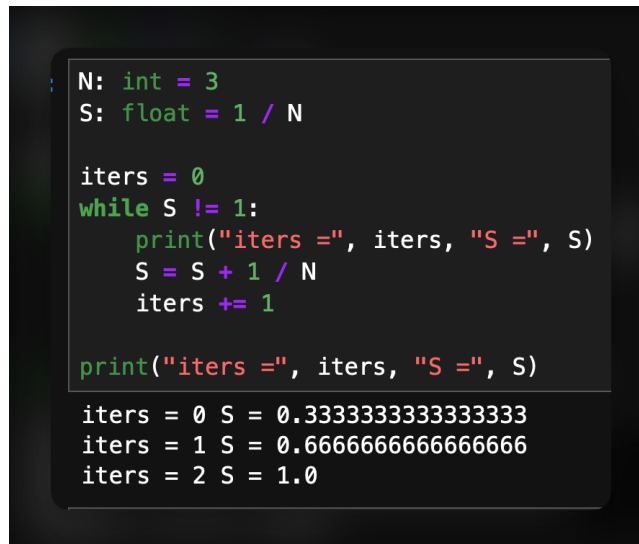
Рис. 6: Выполнение на C++

Вывод. При корректном выборе одинаковой точности (Pascal: single, C++: float) существенной разницы в поведении программ нет: оба языка выполняют вещественное деление и сложение в указанном формате, и последовательность значений совпадает. Точность представления: формат IEEE-754 single (32 бита) имеет 24 бита значащих, что соответствует $\log_{10}(2^{24}) \approx 7,22$ значащим десятичным цифрам — то есть примерно 7 значащих десятичных цифр. (Поэтому видимо и выводится именно 3.333333433E-01 — 4, которая стоит на месте 3 как раз 7 цифра после запятой)

Перевод на Python и сравнение

Переведите программу на язык Python. Какие результаты получены? Сравните их с результатами выполнения программы на C++.

```
1 N: int = 3
2 S: float = 1 / N
3
4 iters = 0
5 while S != 1:
6     print("iters =", iters, "S =", S)
7     S = S + 1 / N
8     iters += 1
9
10 print("iters =", iters, "S =", S)
```



```
: N: int = 3
  S: float = 1 / N

iters = 0
while S != 1:
    print("iters =", iters, "S =", S)
    S = S + 1 / N
    iters += 1

print("iters =", iters, "S =", S)

iters = 0 S = 0.3333333333333333
iters = 1 S = 0.6666666666666666
iters = 2 S = 1.0
```

Рис. 7: Выполнение на Python

Сравнение. Результаты выполнения программы на python ничем не отличаются от результатов выполнения программы на C++, за исключением точности. В python не такая гранулированная система типов чисел как в C++ – в стандартном python float – двухкратная точность, а int теоретически вообще может расти до бесконечности, потому что реализован через массив более маленьких фиксированного размера. На сколько я помню Integer в Haskell имеет такую же идею (в сравнении с Int)

Проверка на других значениях N

Как изменятся результаты, если организовать ввод значений переменной N и выполнить программу для других значений? Проведите эксперименты с разными значениями (7, 11). Всегда ли цикл будет выполняться конечное число раз? Чем объясняются полученные результаты? Опишите свои эксперименты в отчёте (заполните трассировочные таблицы, как это было показано выше) и поясните полученные ответы (коды вещественных чисел – их внутреннее представление – можно посмотреть в режиме отладки в 16-ричной системе, переключившись на дизассемблированный код, ...).

Оператор/ операция	Десятичное значение S: ожидаемое/полученное	Внутреннее представление S	Комментарий
unsigned char N = 7;	—	—	Инициализация
S = 1.0f / N;	0.14285715/0.14285715	0x3E124925	1/7 округлѐн до ближайшего representable
S != 1.0f	истина	0x3E124925	Условие цикла выполнено
S = S + 1.0f / N;	0.2857143/0.2857143	0x3E924925	Сумма 2 аппроксимаций 1/7
S != 1.0f	истина	0x3E924925	Условие цикла выполнено
S = S + 1.0f / N;	0.42857146/0.42857146	0x3EDB6DB8	Сумма 3 аппроксимаций 1/7
S != 1.0f	истина	0x3EDB6DB8	Условие цикла выполнено
S = S + 1.0f / N;	0.5714286/0.5714286	0x3F124925	Сумма 4 аппроксимаций 1/7
S != 1.0f	истина	0x3F124925	Условие цикла выполнено
S = S + 1.0f / N;	0.71428573/0.71428573	0x3F36DB6E	Сумма 5 аппроксимаций 1/7
S != 1.0f	истина	0x3F36DB6E	Условие цикла выполнено
S = S + 1.0f / N;	0.85714287/0.85714287	0x3F5B6DB7	Сумма 6 аппроксимаций 1/7
S != 1.0f	истина	0x3F5B6DB7	Условие цикла выполнено
S = S + 1.0f / N;	1.0/1.0	0x3F800000	Округление даёт ровно 1.0
S != 1.0f	ложь	0x3F800000	Выход из цикла

Оператор/ операция	Десятичное значение S: ожидаемое/полученное	Внутреннее представление S	Комментарий
$S = 1.0f / N;$	0.09090909/0.09090909	0x3DBA2E8C	1/11 округлѐн до ближайшего
$S \neq 1.0f$	истина	0x3DBA2E8C	Условие цикла выполнено
$S = S + 1.0f / N;$	0.18181819/0.18181819	0x3E3A2E8C	Сумма 2 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3E3A2E8C	Условие цикла выполнено
$S = S + 1.0f / N;$	0.27272728/0.27272728	0x3E8BA2E9	Сумма 3 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3E8BA2E9	Условие цикла выполнено
$S = S + 1.0f / N;$	0.36363637/0.36363637	0x3EBA2E8C	Сумма 4 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3EBA2E8C	Условие цикла выполнено
$S = S + 1.0f / N;$	0.45454547/0.45454547	0x3EE8BA2F	Сумма 5 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3EE8BA2F	Условие цикла выполнено
$S = S + 1.0f / N;$	0.54545456/0.54545456	0x3F0BA2E9	Сумма 6 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3F0BA2E9	Условие цикла выполнено
$S = S + 1.0f / N;$	0.63636363/0.63636363	0x3F22E8BA	Сумма 7 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3F22E8BA	Условие цикла выполнено
$S = S + 1.0f / N;$	0.72727275/0.72727275	0x3F3A2E8C	Сумма 8 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3F3A2E8C	Условие цикла выполнено
$S = S + 1.0f / N;$	0.81818187/0.81818187	0x3F51745E	Сумма 9 аппрок- симаций 1/11
$S \neq 1.0f$	истина	0x3F51745E	Условие цикла выполнено
$S = S + 1.0f / N;$	0.909091/0.909091	0x3F68BA30	Сумма 10 ап- проксимаций 1/11
$S \neq 1.0f$	истина	0x3F68BA30	Условие цикла выполнено
$S = S + 1.0f / N;$	1.0/1.00000012	0x3F800001	Ожидаемо 1, но округление даёт 1+ULP
$S \neq 1.0f$	истина	0x3F800001	Цикл не завер- шится

1. Всегда ли цикл выполнится конечное число раз? Нет. Зависит от того, попадёт ли накопленное значение S ровно в представимое $1.0f$. `float` (IEEE-754 single) имеет конечную точность. Значение $1/N$ часто не представимо точно, при суммировании идут округления; последовательность $S_k = \text{round}(k \cdot (1/N))$ в типе `float` может в какой-то момент дать ровно $1.0f$ (тогда цикл закончится) или никогда не дать ровно $1.0f$ (тогда цикл не закончится).

- $N = 3$ — завершается (после трёх сумм S достигает ровно $1.0f$).
- $N = 7$ — завершается (после семи сумм S округляется к $1.0f$).
- $N = 11$ — НЕ завершается: на шаге, где ожидалось ровно 1, получается $1.00000012f = 0x3F800001$, т.е. не ровно $1.0f$, и условие $S \neq 1.0f$ остаётся истинным.

1. Чем объясняются полученные результаты? В типе `float` представимые числа расположены неравномерно: между ними есть шаг — ULP (“unit in last place”), который увеличивается при росте самого числа. Все операции сложения округляют результат к ближайшему представимому значению, поэтому выражение $k \cdot (1/N)$ в памяти хранится не как точная дробь, а как ближайший `float`. В одних случаях накопленные округления дают ровно $1.0f$, и тогда цикл завершается. В других случаях сумма либо становится чуть больше 1, либо приращение становится меньше полу-ULP и значение перестаёт изменяться, поэтому условие $S \neq 1.0f$ остаётся истинным и цикл бесконечный. Например, при $N = 11$ вместо точного значения $1.0f$ получается примерно 1.00000012 и цикл не завершается.