ПИН-22, команда 5: Надёжкина Анна, Трусов Михаил

Отчет

Лабораторная работа 3 (0011 = 3)

Арифметика в ЭВМ и представление данных (целочисленные операции)

Операционная система ОС Windows, 64-разрядная ОС.

Задание Л3.з1. *(Вариант 2 m=33 n=-101)*

Разработайте программу на языке C++, которая расширяет значение целочисленной переменной из 16 бит до 32 бит, рассматривая числа как: – знаковые (signed); – беззнаковые (unsigned). Проверьте её работу на значениях m и n (таблица Л3.1). Каждое из двух значений — как m, так и n — должно расширяться двумя способами — как знаковым, так и беззнаковым (итого четыре операции).

Бонус +1 балл, если младшая цифра каждого представления print32() находится под младшей цифрой соответствующего представления print16().

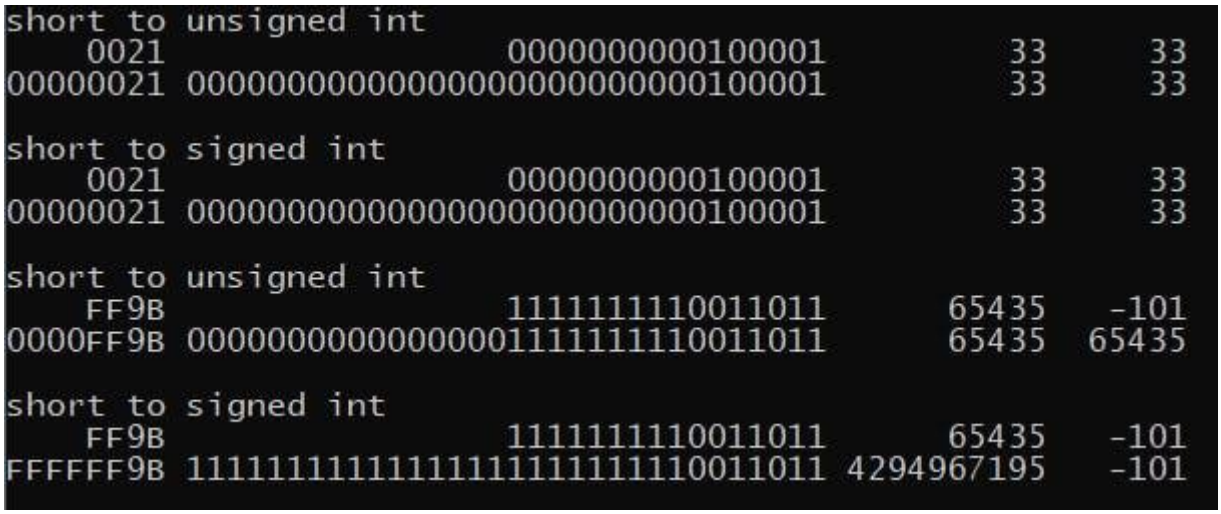
Программный код:

```
#include <stdio.h>

#include <bitset>

using namespace std;

void print16(void *p) {
  unsigned short *a = reinterpret_cast<unsigned short *>(p);
  printf("    ");
  printf("%04X ", *a);
  printf("%32s ", bitset<16>(*a).to_string().c_str());
  printf("%10u ", *a);

  short *b = reinterpret_cast<short *>(p);
  printf("%6d \n", *b);
}

void print32(void *p) {
```

```
  unsigned int *a = reinterpret_cast<unsigned int *>(p);
  printf("%08X ", *a);
  printf("%32s ", bitset<32>(*a).to_string().c_str());
  printf("%10u ", *a);

  int *b = reinterpret_cast<int *>(p);
  printf("%6d \n\n", *b);
}

int main() {
  printf("short to unsigned int\n");
  unsigned short m1 = 33;
  unsigned int m11 = static_cast<unsigned int>(m1);
  print16(&m1);
  print32(&m11);

  printf("short to signed int\n");
  short m2 = 33;
  int m22 = static_cast<int>(m2);
  print16(&m2);
  print32(&m22);

  printf("short to unsigned int\n");
  unsigned short n1 = -101;
  unsigned int n11 = static_cast<unsigned int>(n1);
  print16(&n1);
  print32(&n11);

  printf("short to signed int\n");
  short n2 = -101;
  int n22 = static_cast<int>(n2);
```

```
  print16(&n2);
  print32(&n22);

  return 0;
}
```

Вывод:



Задание Л3.з2.

Разработайте программу на языке C/C++, которая выполняет над 16-битным целочисленным значением x:

– знаковое умножение на 2;

– беззнаковое умножение на 2;

– знаковое деление на 2;

– беззнаковое деление на 2;

– расчёт остатка от беззнакового деления на 16;

– округление вниз до числа, кратного 16 (беззнаковое). а также:

– знаковый сдвиг влево на 1 бит;

– беззнаковый сдвиг влево на 1 бит;

– знаковый сдвиг вправо на 1 бит;

– беззнаковый сдвиг вправо на 1 бит;

– рассчитывает x & 15;

– рассчитывает x & −16.

Сопоставьте результаты — вначале на значении m, затем n (таблица Л3.1)

Программный код:

```
#include <stdio.h>
#include <bitset>

using namespace std;


void print16(void *p) {
    unsigned short *a = reinterpret_cast<unsigned short *>(p);
    printf("%04X ", *a);
    printf("%16s ", bitset<16>(*a).to_string().c_str());
    printf("unsigned: %5u ", *a);

    short *b = reinterpret_cast<short *>(p);
    printf("signed: %6d \n", *b);
}



int main() {

    short m = 33;
    printf("m = 33\n");
    printf("Znakovoe umnozhenie na 2\n");
    short count = 2;
    short ml = *reinterpret_cast<short*>(&m);
    short resml = ml * count;
    print16(&ml);
    print16(&count);
    print16(&resml);


    printf("\nBezznakovoe umnozhenie na 2\n");
```

```
unsigned short count2 = 2;
unsigned short m2 = *reinterpret_cast<unsigned short*>(&m);
unsigned short resm2 = m2 * count2;
printl6(&m2);
printl6(&count2);
printl6(&resm2);


printf("\nZnakovoe delenie na 2\n");
short count3 = 2;
short m3 = *reinterpret_cast<short*>(&m);
short resm3 = m3 / count3;
printl6(&m3);
printl6(&count3);
printl6(&resm3);


printf("\nBezznakovoe delenie na 2\n");
unsigned short count4 = 2;
unsigned short m4 = *reinterpret_cast<unsigned short*>(&m);
unsigned short resm4 = m4 / count4;
printl6(&m4);
printl6(&count4);
printl6(&resm4);

printf("\nOstatok ot delenia na 16\n");
unsigned short count5 = 16;
unsigned short m5 = *reinterpret_cast<unsigned short*>(&m);
unsigned short resm5 = m5 % count5;
printl6(&m5);
printl6(&count5);
printl6(&resm5);
```

```
printf("\n0kruglenie vniz do chisla, kratnogo 16\n");
unsigned short count6 = 16;
unsigned short m6 = *reinterpret_cast<unsigned short*>(&m);
unsigned short resm6 = (m6 / count6) * count6;
printl6(&m6);
printl6(&count6);
printl6(&resm6);

short n = -101;

printf("\nn = -101\n");
printf("Znakovoe umnozhenie na 2\n");
short nl = *reinterpret_cast<short*>(&n);
short resnl = nl * countl;
printl6(&nl);
printl6(&countl);
printl6(&resnl);


printf("\nBezznakovoe umnozhenie na 2\n");
unsigned short n2 = *reinterpret_cast<unsigned short*>(&n);
unsigned short resn2 = n2 * count2;
printl6(&n2);
printl6(&count2);
printl6(&resn2);


printf("\nZnakovoe delenie na 2\n");
short n3 = *reinterpret_cast<short*>(&n);
short resn3 = n3 / count3;
printl6(&n3);
printl6(&count3);
```

```
    printf16(&resn3);



    printf("\nBezznakovoe delenie na 2\n");
    unsigned short n4 = *reinterpret_cast<unsigned short*>(&n);
    unsigned short resn4 = n4 / count4;
    printf16(&n4);
    printf16(&count4);
    printf16(&resn4);

    printf("\nOstatok ot delenia na 16\n");
    unsigned short n5 = *reinterpret_cast<unsigned short*>(&n);
    unsigned short resn5 = n5 % count5;
    printf16(&n5);
    printf16(&count5);
    printf16(&resn5);



    printf("\nOkruglenie vniz do chisla, kratnogo 16\n");
    unsigned short n6 = *reinterpret_cast<unsigned short*>(&n);
    unsigned short resn6 = (n6 / count6) * count6;
    printf16(&n6);
    printf16(&count6);
    printf16(&resn6);




    return 0;
}
```

Вывод:

```
m = 33
Znakovoe umnozhenie na 2
0021 0000000000100001 unsigned:    33 signed:     33
0002 0000000000000010 unsigned:     2 signed:      2
0042 0000000001000010 unsigned:    66 signed:     66

Bezznakovoe umnozhenie na 2
0021 0000000000100001 unsigned:    33 signed:     33
0002 0000000000000010 unsigned:     2 signed:      2
0042 0000000001000010 unsigned:    66 signed:     66

Znakovoe delenie na 2
0021 0000000000100001 unsigned:    33 signed:     33
0002 0000000000000010 unsigned:     2 signed:      2
0010 0000000000010000 unsigned:    16 signed:     16

Bezznakovoe delenie na 2
0021 0000000000100001 unsigned:    33 signed:     33
0002 0000000000000010 unsigned:     2 signed:      2
0010 0000000000010000 unsigned:    16 signed:     16

Ostatok ot delenia na 16
0021 0000000000100001 unsigned:    33 signed:     33
0010 0000000000010000 unsigned:    16 signed:     16
0001 0000000000000001 unsigned:     1 signed:      1

Okruglenie vniz do chisla, kratnogo 16
0021 0000000000100001 unsigned:    33 signed:     33
0010 0000000000010000 unsigned:    16 signed:     16
0020 0000000000100000 unsigned:    32 signed:     32

n = -101
Znakovoe umnozhenie na 2
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0002 0000000000000010 unsigned:     2 signed:      2
FF36 1111111100110110 unsigned: 65334 signed:   -202

Bezznakovoe umnozhenie na 2
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0002 0000000000000010 unsigned:     2 signed:      2
FF36 1111111100110110 unsigned: 65334 signed:   -202

Znakovoe delenie na 2
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0002 0000000000000010 unsigned:     2 signed:      2
FFCE 1111111111001110 unsigned: 65486 signed:    -50

Bezznakovoe delenie na 2
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0002 0000000000000010 unsigned:     2 signed:      2
7FCD 0111111111001101 unsigned: 32717 signed:  32717

Ostatok ot delenia na 16
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0010 0000000000010000 unsigned:    16 signed:     16
000B 0000000000001011 unsigned:    11 signed:     11

Okruglenie vniz do chisla, kratnogo 16
FF9B 1111111110011011 unsigned: 65435 signed:   -101
0010 0000000000010000 unsigned:    16 signed:     16
FF90 1111111110010000 unsigned: 65424 signed:   -112
```

Программный код

```
#include <stdio.h>

#include <bitset>

using namespace std;

void print16(void *p) {
  unsigned short *a = reinterpret_cast<unsigned short *>(p);
  printf("%04X ", *a);
  printf("%16s ", bitset<16>(*a).to_string().c_str());
  printf("unsigned: %5u ", *a);

  short *b = reinterpret_cast<short *>(p);
  printf("signed: %6d \n", *b);
}

int main() {
  short m = 33;
  printf("m = 33\n");
  printf("Znakovyj sdvig vlevo na 1 bit\n");
  short m1 = *reinterpret_cast<short *>(&m);
  short resm1 = m1 << 1;
  print16(&m1);
  print16(&resm1);

  printf("\nBezznakovyj sdvig vlevo na 1 bit\n");
  unsigned short m2 = *reinterpret_cast<unsigned short *>(&m);
  short resm2 = m2 << 1;
  print16(&m2);
  print16(&resm2);

  printf("\nZnakovyj sdvig vpravo na 1 bit\n");
```

```
short m3 = *reinterpret_cast<short *>(&m);
short resm3 = m3 >> 1;
printl6(&m3);
printl6(&resm3);

printf("\nBezznakovyj sdvig vpravo na 1 bit\n");
unsigned short m4 = *reinterpret_cast<unsigned short *>(&m);
short resm4 = m4 >> 1;
printl6(&m4);
printl6(&resm4);

printf("\nPobitovoe AND 15\n");
unsigned short m5 = *reinterpret_cast<unsigned short *>(&m);
unsigned short count5 = 15;
unsigned short resm5 = m5 & count5;
printl6(&m5);
printl6(&count5);
printl6(&resm5);

printf("\nPobitovoe AND -16\n");
short m6 = *reinterpret_cast<short *>(&m);
short count6 = -16;
short resm6 = m6 & count6;
printl6(&m6);
printl6(&count6);
printl6(&resm6);

short n = -101;
printf("\nn = -101\n");
printf("Znakovyj sdvig vlevo na 1 bit\n");
short nl = *reinterpret_cast<short *>(&n);
short resnl = nl << 1;
printl6(&nl);
```

```
printl6(&resn1);

printf("\nBezznakovyj sdvig vlevo na 1 bit\n");
unsigned short n2 = *reinterpret_cast<unsigned short *>(&n);
short resn2 = n2 << 1;
printl6(&n2);
printl6(&resn2);

printf("\nZnakovyj sdvig vpravo na 1 bit\n");
short n3 = *reinterpret_cast<short *>(&n);
short resn3 = n3 >> 1;
printl6(&n3);
printl6(&resn3);

printf("\nBezznakovyj sdvig vpravo na 1 bit\n");
unsigned short n4 = *reinterpret_cast<unsigned short *>(&n);
short resn4 = n4 >> 1;
printl6(&n4);
printl6(&resn4);

printf("\nPobitovoe AND 15\n");
unsigned short n5 = *reinterpret_cast<unsigned short *>(&n);
unsigned short resn5 = n5 & count5;
printl6(&n5);
printl6(&count5);
printl6(&resn5);

printf("\nPobitovoe AND -16\n");
short n6 = *reinterpret_cast<short *>(&n);
short resn6 = n6 & count6;
printl6(&n6);
printl6(&count6);
printl6(&resn6);
```

```
 return 0;
}
```

Вывод:

```
m = 33
Znakovyj sdvig vlevo na 1 bit
0021 0000000000100001 unsigned:    33 signed:      33
0042 0000000001000010 unsigned:    66 signed:      66

Bezznakovyj sdvig vlevo na 1 bit
0021 0000000000100001 unsigned:    33 signed:      33
0042 0000000001000010 unsigned:    66 signed:      66

Znakovyj sdvig vpravo na 1 bit
0021 0000000000100001 unsigned:    33 signed:      33
0010 0000000000010000 unsigned:    16 signed:      16

Bezznakovyj sdvig vpravo na 1 bit
0021 0000000000100001 unsigned:    33 signed:      33
0010 0000000000010000 unsigned:    16 signed:      16

Pobitovoe AND 15
0021 0000000000100001 unsigned:    33 signed:      33
000F 0000000000001111 unsigned:    15 signed:      15
0001 0000000000000001 unsigned:     1 signed:       1

Pobitovoe AND -16
0021 0000000000100001 unsigned:    33 signed:      33
FFF0 1111111111110000 unsigned: 65520 signed:     -16
0020 0000000000100000 unsigned:    32 signed:      32

n = -101
Znakovyj sdvig vlevo na 1 bit
FF9B 1111111110011011 unsigned: 65435 signed:    -101
FF36 1111111100110110 unsigned: 65334 signed:    -202

Bezznakovyj sdvig vlevo na 1 bit
FF9B 1111111110011011 unsigned: 65435 signed:    -101
FF36 1111111100110110 unsigned: 65334 signed:    -202

Znakovyj sdvig vpravo na 1 bit
FF9B 1111111110011011 unsigned: 65435 signed:    -101
FFCD 1111111111001101 unsigned: 65485 signed:     -51

Bezznakovyj sdvig vpravo na 1 bit
FF9B 1111111110011011 unsigned: 65435 signed:    -101
7FCD 0111111111001101 unsigned: 32717 signed:   32717

Pobitovoe AND 15
FF9B 1111111110011011 unsigned: 65435 signed:    -101
000F 0000000000001111 unsigned:    15 signed:      15
000B 0000000000001011 unsigned:    11 signed:      11

Pobitovoe AND -16
FF9B 1111111110011011 unsigned: 65435 signed:    -101
FFF0 1111111111110000 unsigned: 65520 signed:     -16
FF90 1111111110010000 unsigned: 65424 signed:    -112
```

Задание Л3.з3. *(Вариант 5 D=256)*

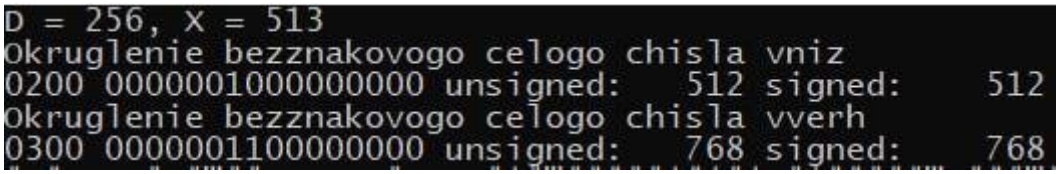
Разработайте программу на языке C/C++, которая, используя только сложение, вычитание и побитовые операции, округляет целочисленное беззнаковое значение x до кратного значению D (таблица Л3.2) двумя способами: а) вниз; б) вверх.

Программный код:

```
#include <stdio.h>


#include <bitset>

using namespace std;


void print16(void *p) {
  unsigned short *a = reinterpret_cast<unsigned short *>(p);
  printf("%04X ", *a);
  printf("%16s ", bitset<16>(*a).to_string().c_str());
  printf("unsigned: %5u ", *a);

  short *b = reinterpret_cast<short *>(p);
  printf("signed: %6d \n", *b);
}


int main() {
  unsigned short d = 256;
  unsigned short x = 513;
  printf("D = 256, X = 513\n");
  printf("Okruglenie bezznakovogo celogo chisla vniz\n");
  unsigned short dminus = d - 1;
  unsigned short ans1 = x & ~dminus;
  print16(&ans1);

  printf("Okruglenie bezznakovogo celogo chisla vverh\n");
  unsigned short not_dminus = ~dminus;
  unsigned short x_plus_dminus = x + dminus;
  unsigned short ans2 = (x_plus_dminus & not_dminus);
  print16(&ans2);
```

```
 return 0;
}
```

Вывод:



Задание Л3.з4. *Вариант 1 $a = 0$, $b = 1$, $c = 12345678$, $d = 123456789$*

Разработайте программу на языке C/C++, которая выполняет для 32-битной переменной x целочисленный инкремент (то есть целочисленная интерпретация x должна увеличиться на 1). Проверьте её работу на 32-битных целочисленных значениях m и n (таблица Л3.1), 32-битных значениях с плавающей запятой a, b, c, d из таблицы Л3.3, а также на целочисленных значениях:

– 0;

– максимальное целое 32-битное значение без знака;

– минимальное целое 32-битное значение со знаком;

– максимальное целое 32-битное значение со знаком

Программный код:

Инкремент:

```
#include <stdio.h>

#include <bitset>
#include <iostream>
using namespace std;

void print32(void *p) {
  unsigned int *a = reinterpret_cast<unsigned int *>(p);
  printf("%08X ", *a);
  printf("%32s ", bitset<32>(*a).to_string().c_str());
  printf("unsigned: %10u ", *a);

  int *b = reinterpret_cast<int *>(p);
  printf("signed: %6d \n", *b);
}
```

```
int main() {
  printf("int m = 33\n");
  int m = 33;
  print32(&m);
  int mi = *reinterpret_cast<int *>(&m) + 1;
  print32(&mi);

  printf("\nint n = -101\n");
  int n = -101;
  print32(&n);
  int ni = *reinterpret_cast<int *>(&n) + 1;
  print32(&ni);

  printf("\nfloat a = 0\n");
  float a = 0;
  print32(&a);
  int ai = *reinterpret_cast<int *>(&a) + 1;
  print32(&ai);

  printf("\nfloat b = 1\n");
  float b = 1;
  print32(&b);
  int bi = *reinterpret_cast<int *>(&b) + 1;
  print32(&bi);

  printf("\nfloat c = 12345678\n");
  float c = 12345678;
  print32(&c);
  int ci = *reinterpret_cast<int *>(&c) + 1;
  print32(&ci);

  printf("\nfloat d = 123456789\n");
```

```
  float d = 123456789;
  print32(&d);
  int di = *reinterpret_cast<int *>(&d) + 1;
  print32(&di);

  printf("\nint zero = 0\n");
  int zero = 0;
  print32(&zero);
  int zeroi = *reinterpret_cast<int *>(&zero) + 1;
  print32(&zeroi);

  printf("\nunsigned int max = 4294967295\n");
  unsigned int max_unsigned_int = 4294967295;
  print32(&max_unsigned_int);
  int max_unsigned_inti = *reinterpret_cast<int *>(&max_unsigned_int) + 1;
  print32(&max_unsigned_inti);

  printf("\nint min_int = -2147483648\n");
  int min_int = -2147483648;
  print32(&min_int);
  int min_inti = *reinterpret_cast<int *>(&min_int) + 1;
  print32(&min_inti);

  printf("\nint max_int = 2147483647\n");
  int max_int = 2147483647;
  print32(&max_int);
  int max_inti = *reinterpret_cast<int *>(&max_int) + 1;
  print32(&max_inti);

  return 0;
}
```

Вывод:

```
int m = 33
00000021 00000000000000000000000000100001 unsigned:          33 signed:      33
00000022 00000000000000000000000000100010 unsigned:          34 signed:      34

int n = -101
FFFFFF9B 11111111111111111111111110011011 unsigned: 4294967195 signed:    -101
FFFFFF9C 11111111111111111111111110011100 unsigned: 4294967196 signed:    -100

float a = 0
00000000 00000000000000000000000000000000 unsigned:           0 signed:       0
00000001 00000000000000000000000000000001 unsigned:           1 signed:       1

float b = 1
3F800000 00111111100000000000000000000000 unsigned: 1065353216 signed: 1065353216
3F800001 00111111100000000000000000000001 unsigned: 1065353217 signed: 1065353217

float c = 12345678
4B3C614E 01001011001111000110000101001110 unsigned: 1262248270 signed: 1262248270
4B3C614F 01001011001111000110000101001111 unsigned: 1262248271 signed: 1262248271

float d = 123456789
4CEB79A3 01001100111010110111100110100011 unsigned: 1290500515 signed: 1290500515
4CEB79A4 01001100111010110111100110100100 unsigned: 1290500516 signed: 1290500516

int zero = 0
00000000 00000000000000000000000000000000 unsigned:           0 signed:       0
00000001 00000000000000000000000000000001 unsigned:           1 signed:       1

unsigned int max = 4294967295
FFFFFFFF 11111111111111111111111111111111 unsigned: 4294967295 signed:      -1
00000000 00000000000000000000000000000000 unsigned:           0 signed:       0

int min_int = -2147483648
80000000 10000000000000000000000000000000 unsigned: 2147483648 signed: -2147483648
80000001 10000000000000000000000000000001 unsigned: 2147483649 signed: -2147483647

int max_int = 2147483647
7FFFFFFF 01111111111111111111111111111111 unsigned: 2147483647 signed: 2147483647
80000000 10000000000000000000000000000000 unsigned: 2147483648 signed: -2147483648
```

Декремент:

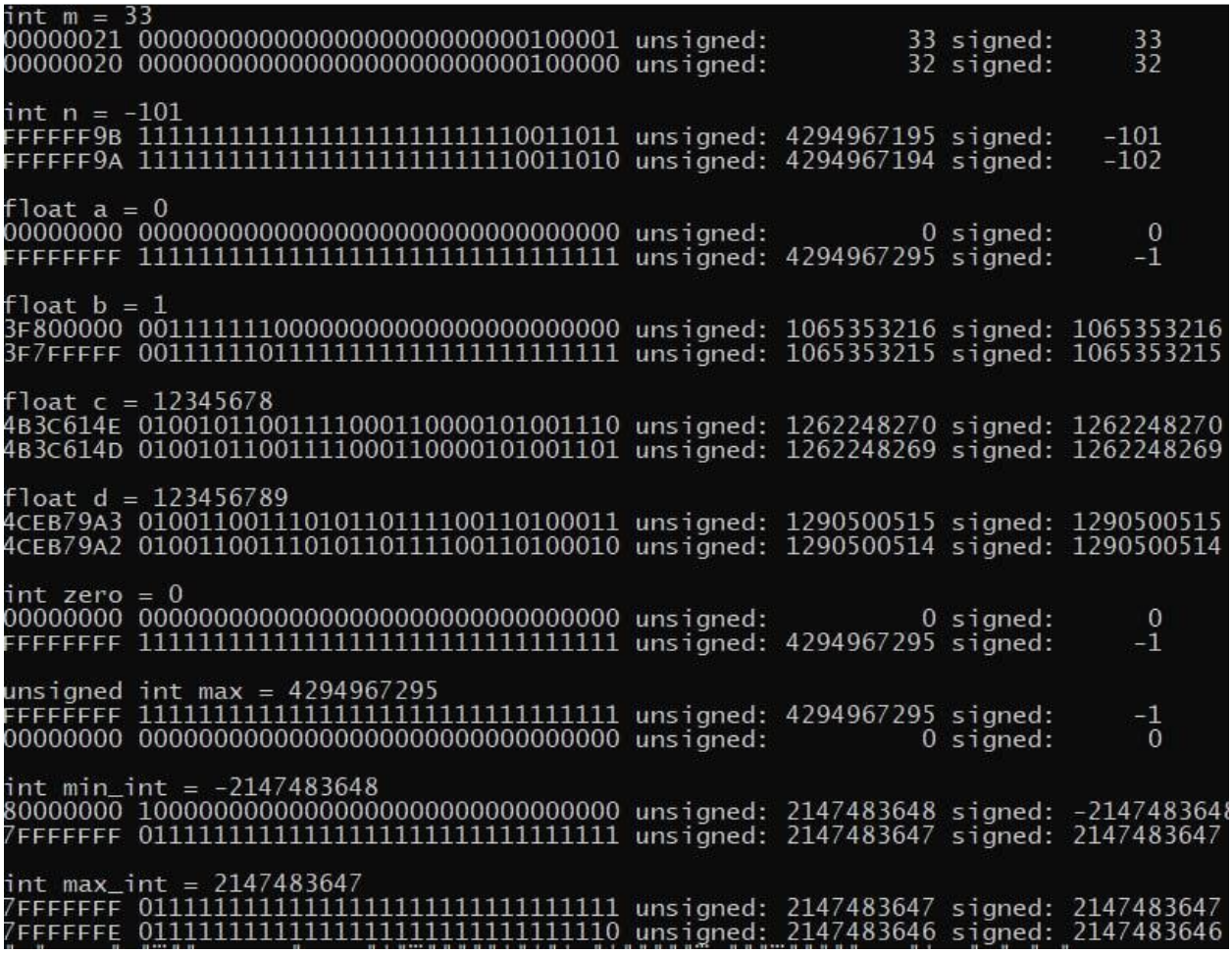
```
#include <stdio.h>

#include <bitset>

#include <iostream>

using namespace std;


void print32(void *p) {

    unsigned int *a = reinterpret_cast<unsigned int *>(p);

    printf("%08X ", *a);

    printf("%32s ", bitset<32>(*a).to_string().c_str());

    printf("unsigned: %10u ", *a);


    int *b = reinterpret_cast<int *>(p);

    printf("signed: %6d \n", *b);

}
```

```
int main() {
    printf("int m = 33\n");
    int m = 33;
    print32(&m);
    int mi = *reinterpret_cast<int*>(&m)-1;
    print32(&mi);

    printf("\nint n = -101\n");
    int n = -101;
    print32(&n);
    int ni = *reinterpret_cast<int*>(&n)-1;
    print32(&ni);


    printf("\nfloat a = 0\n");
    float a = 0;
    print32(&a);
    int ai = *reinterpret_cast<int*>(&a)-1;
    print32(&ai);

    printf("\nfloat b = 1\n");
    float b = 1;
    print32(&b);
    int bi = *reinterpret_cast<int*>(&b)-1;
    print32(&bi);

    printf("\nfloat c = 12345678\n");
    float c = 12345678;
    print32(&c);
    int ci = *reinterpret_cast<int*>(&c)-1;
    print32(&ci);
```

```
printf("\nfloat d = 123456789\n");
float d = 123456789;
print32(&d);
int di = *reinterpret_cast<int*>(&d)-1;
print32(&di);


printf("\nint zero = 0\n");
int zero = 0;
print32(&zero);
int zeroi = *reinterpret_cast<int*>(&zero)-1;
print32(&zeroi);

printf("\nunsigned int max = 4294967295\n");
unsigned int max_unsigned_int = 4294967295;
print32(&max_unsigned_int);
int max_unsigned_inti = *reinterpret_cast<int*>(&max_unsigned_int)+1;
print32(&max_unsigned_inti);

printf("\nint min_int = -2147483648\n");
int min_int = -2147483648;
print32(&min_int);
int min_inti = *reinterpret_cast<int*>(&min_int)-1;
print32(&min_inti);

printf("\nint max_int = 2147483647\n");
int max_int = 2147483647;
print32(&max_int);
int max_inti = *reinterpret_cast<int*>(&max_int)-1;
print32(&max_inti);

return 0;
```

}

Вывод:



Задание Л3.з5.

Рассчитайте для заданного 32-битного значения с плавающей запятой x его модуль $|x|$, используя только битовые (целочисленные) операции и преобразование указателей

Программный код

```
#include <stdio.h>


#include <bitset>

#include <iostream>

using namespace std;


void print32(void *p) {

  unsigned int *a = reinterpret_cast<unsigned int *>(p);
```

```
  printf("%08X ", *a);
  printf("%32s ", bitset<32>(*a).to_string().c_str());

  float *c = reinterpret_cast<float *>(p);
  printf("%5.2f ", *c);
  printf("%5.2e \n", *c);
}

float calculateAbsolute(float x) {
  int *xPtr = reinterpret_cast<int *>(&x);
  *xPtr &= 0x7FFFFFFF;
  return *reinterpret_cast<float *>(xPtr);
}

int main() {
  float x = -2.0 / 3.0;
  print32(&x);
  float y = calculateAbsolute(x);
  print32(&y);
  return 0;
}
```

Вывод:

Задание Л3.з6.

Разработайте программу на языке C/C++, выполняющую вычисления над числами с плавающей запятой одинарной точности (float). Проверьте, что программа действительно работает с операндами одинарной точности, а не приводит к типу float окончательный результат. Для частичной суммы гармонического ряда $S(N) = \sum_{i=1}^{N} 1/i \in R$ найдите две её оценки: $Sd(N)$ — последовательно складывая члены, начиная от $i = 1$ и заканчивая $i = N$ («наивный» порядок), и $Sa(N)$ — от $i = N$ к $i = 1$. Сравните $Sd(N)$ и $Sa(N)$ для различных значений N: 103 , 106 , 109 . Объясните результат. Измените тип операндов на double. Для вывода результата используйте print64() (если она не реализована — обязательно выведите, кроме double-интерпретации, также и long long-интерпретацию результата в шестнадцатеричном представлении). Объясните результат

Программный код

```
#include <stdio.h>

#include <bitset>
#include <iostream>
using namespace std;

void print32(void *p) {
    unsigned int *a = reinterpret_cast<unsigned int *>(p);
    printf("%08X ", *a);
    printf("%32s ", bitset<32>(*a).to_string().c_str());

    float *c = reinterpret_cast<float *>(p);
    printf("%5.2f ", *c);
    printf("%5.2e \n", *c);
}

void print64(void *p) {
    unsigned long long *a = reinterpret_cast<unsigned long long *>(p);
    printf("%016X ", *a);
    printf("%64s ", bitset<64>(*a).to_string().c_str());

    double *c = reinterpret_cast<double *>(p);
    printf("%5.2f ", *c);
    printf("%5.2e \n", *c);
}

void getSumFloat(int N) {
    printf("naiveSum:\n");
    float naiveSum = 0.0;
    for (int i = 1; i <= N; i++) {
        naiveSum += (1.0 / i);
    }
```

```
    print32(&naiveSum);

    printf("reversedSum:\n");
    float reversedSum = 0;
    for (int i = N; i >= 1; i--) {
        reversedSum += (1.0 / i);
    }
    print32(&reversedSum);
}

void getSumDouble(int N) {
    printf("naiveSum:\n");
    double naiveSum = 0.0;
    for (int i = 1; i <= N; i++) {
        naiveSum += (1.0 / i);
    }

    print64(&naiveSum);

    printf("reversedSum:\n");
    double reversedSum = 0;
    for (int i = N; i >= 1; i--) {
        reversedSum += (1.0 / i);
    }
    print64(&reversedSum);
}

int main() {
    printf("float, N = 1000\n");
    getSumFloat(1000);
    printf("\nfloat, N = 1000000\n");
    getSumFloat(1000000);
    printf("\nfloat, N = 1000000000\n");
```

```
    getSumFloat(10000000000);


    printf("\n\ndouble, N = 1000\n");
    getSumDouble(1000);
    printf("\ndouble, N = 1000000\n");
    getSumDouble(1000000);
    printf("\ndouble, N = 1000000000\n");
    getSumDouble(1000000000);
}
```

Вывод

```
float, N = 1000
naiveSum:
40EF890A 01000000111011111000100100001010  7.49 7.49e+00
reversedSum:
40EF88FC 01000000111011111000100011111100  7.49 7.49e+00

float, N = 1000000
naiveSum:
4165B7BD 01000001011001011011011110111101 14.36 1.44e+01
reversedSum:
4166484D 01000001011001100100100001001101 14.39 1.44e+01

float, N = 1000000000
naiveSum:
4176757C 01000001011101100111010101111100 15.40 1.54e+01
reversedSum:
4196769E 01000001100101100111011010011110 18.81 1.88e+01


double, N = 1000
naiveSum:
0000000045F4E618 0100000000011101111100010001111101000101111101001110011000011000  7.49 7.49e+00
reversedSum:
0000000045F4E615 0100000000011101111100010001111101000101111101001110011000010101  7.49 7.49e+00

double, N = 1000000
naiveSum:
000000007A1DF0D6 0100000000101100110010010001001101111010000111011111000011010110 14.39 1.44e+01
reversedSum:
000000007A1DF28F 0100000000101100110010010001001101111010000111011111001010001111 14.39 1.44e+01

double, N = 1000000000
naiveSum:
000000005B11A3D5 0100000000110101010011001110110001011011000100011010001111010101 21.30 2.13e+01
reversedSum:
000000005B11A131 0100000000110101010011001110110001011011000100011010000100110001 21.30 2.13e+01
```