**Задание Л1.з1**

ОС: Windows 10 Pro

Разрядность: 64-разрядная операционная система, процессор x64

Компилятор и его версия: GCC 11.2.0

Разрядность сборки: х64

Архитектура процессора, назначение платформы: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz, платформа общего назначения

**Задание Л1.з2**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

cout << sizeof(char) << endl;

cout<<"Signed char "<<sizeof(signed char)<<endl;

cout<<"unsigned char "<<sizeof(unsigned char)<<endl;

cout<<"wchar\_t "<<sizeof(wchar\_t)<<endl;

cout<<"Short "<<sizeof(short)<<endl;

cout<<"unsigned short "<<sizeof(unsigned short)<<endl;

cout<<"int "<<sizeof(int)<<endl;

cout << sizeof(unsigned int) << endl;

cout << sizeof(long) << endl;

cout << sizeof(unsigned long) << endl;

cout << sizeof(long long) << endl;

cout << sizeof(unsigned long long) << endl;

cout << sizeof(float) << endl;

cout << sizeof(double) << endl;

cout << sizeof(long double) << endl;

cout << sizeof(size\_t) << endl;

cout << sizeof(ptrdiff\_t) << endl;

cout << sizeof(void\*) << endl;

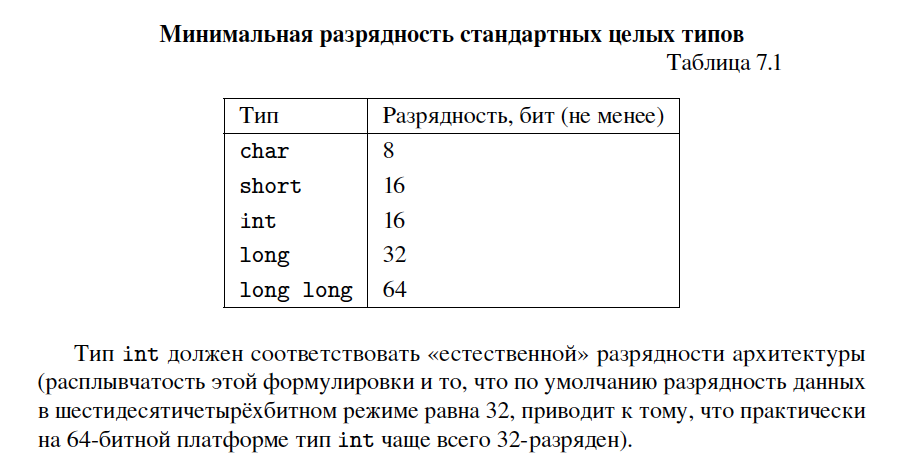
cout << sizeof(char\*) << endl;

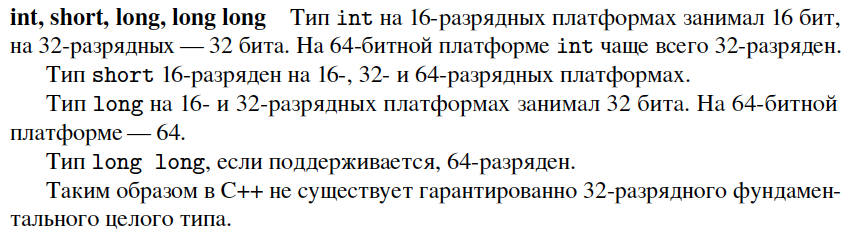
cout << sizeof( int\*) << endl;

cout << sizeof(unsigned int\*) << endl;

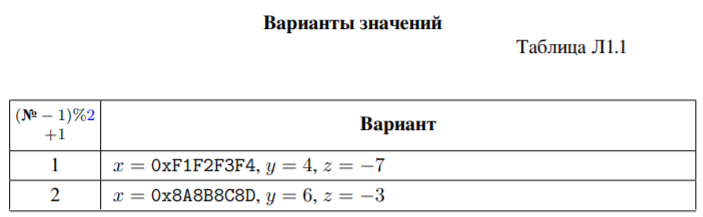
}

| char | 1 |
| --- | --- |
| signed char | 1 |
| unsigned char | 1 |
| wchar\_t | 2 |
| short(разрядность 16 бит) | 2 |
| unsigned short(разрядность 16 бит) | 2 |
| int(разрядность 16 бит) | 4 |
| unsigned int(разрядность 32 бит) | 4 |
| long(разрядность 32 бит на 16- и 32- разрядных платформах и 64-разряден на 64-разрядных) | 4 |
| unsigned long(разрядность 32 бит) | 4 |
| long long(разрядность 64 бит) | 8 |
| unsigned long long(разрядность 64 бит) | 8 |
| float(разрядность 32 бит) | 4 |
| double(разрядность 64 бит) | 8 |
| long double | 16 |
| size\_t(разрядность 64 бит) | 8 |
| ptrdiff\_t(разрядность 64 бит) | 8 |
| void\*(разрядность 64 бит) | 8 |
| char\* | 8 |
| int\*(разрядность 64 бит) | 8 |
| unsigned int\*(разрядность 64 бит) | 8 |





**Задание Л1.з3.**



**Интерпретация одной и той же области памяти при рассмотрении ее как знакового и беззнакового числа.**

На каждое число, относящееся к разным типам данным в памяти выделяется строго определенное число битов, которое не может увеличиваться в зависимости от желания программиста(если на тип данных “unsigned char”(иное название “unsigned \_int8”) выделено только 8 бит и с помощью данного типа можно записать число от 0 до 255 включительно, то изменить данные значения нельзя). Однако, если с беззнаковым представлением все предельно ясно, то со знаковым возникают некоторые сложности, которые были решены с помощью выделения одного из выделенных битов под знак. Таким образом, диапазон значений для знакового аналога “unsigned char”, имеющего название “\_int8”(иное название “char”), сдвигается на половину в отрицательную область и становится равным от -128 до 127. Количество чисел, которые можно закодировать с помощью данного типа остается неизменным, однако существенно меняется и расширяется область его применения.

| **Значения** | **а), г) Целочисленная беззнаковая интерпретация в шестнадцатеричном представлении** | **б), д)Целочисленная беззнаковая интерпретация в двоичном представлении;** | **в)Целочисленная беззнаковая интерпретация в десятичном представлении;** | **е) Целочисленная знаковая интерпретация в десятичном представлении** |
| --- | --- | --- | --- | --- |
| **0** | 0 | 00000000 | 0 | 0 |
| **65535** | ffff | 11111111 | 65535 | -1 |
| **-32768** | 8000 | 00000000 | 32768 | -32768 |
| **32767** | 7fff | 11111111 | 32767 | 32767 |
| **6** | 6 | 00000110 | 6 | 6 |
| **-3** | fffd | 11111101 | 65533 | -3 |

Код программы

#include <iostream>

#include <cstdlib>

#include <array>

#include <bitset>

#include <windows.h>

using namespace std;

void print16(void \*p)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout<<"Цел. без знак. в 16-й с-ме:"<<hex<<\*(reinterpret\_cast<unsigned short\*>(p))<<endl;

cout<<"Цел. без знак. в 2-й с-ме:"<< bitset<8>{\*(reinterpret\_cast<unsigned short\*>(p))}<<endl;

cout<<"Цел. без знак. в 10-й с-ме:"<< dec<<\*(reinterpret\_cast<unsigned short\*>(p))<<endl;

cout<<"Цел. знак. в 16-й с-ме:"<<hex<< \*(reinterpret\_cast<short\*>(p))<<endl;

cout<<"Цел. знак. в 2-й с-ме:"<<bitset<8>{\*(reinterpret\_cast<short\*>(p))}<<endl;

cout<<"Цел. знак. в 10-й с-ме:"<< std::dec<< \*(reinterpret\_cast<short\*>(p))<<endl;

}

int main()

{

unsigned short min16 = 0; //Минимальное целое 16-битное значение без знака

unsigned short max16 = 65535; //Максимальное целое 16-битное значение без знака

short min\_sign = -32768; //Минимальное целое 16-битное значение со знаком

short max\_sign = 32767; //Максимальное целое 16-битное значение со знаком

short y = 6;

short z = -3;

cout<<min16<<endl;

print16(&min16);

cout<<max16<<endl;

print16(&max16);

cout<<min\_sign<<endl;

print16(&min\_sign);

cout<<max\_sign<<endl;

print16(&max\_sign);

cout<<y<<endl;

print16(&y);

cout<<z<<endl;

print16(&z);

return 0;

}

**Задание Л1.з4.**

Для типа float

| **Значение** | **ж) интерпретацию с плавающей запятой в представлении с фиксированным количеством цифр после запятой** | **з) интерпретацию с плавающей запятой в экспоненциальном представлении.** |
| --- | --- | --- |
| 0 | 0.000000 | 0.000000e+00 |
| 4294967295 | nan | nan |
| -2147483648 | -0.000000 | -0.000000e+00 |
| 2147483647 | nan | nan |
| int 0x8A8B8C8D | -0.000000 | -1.343808e-32 |
| int 6 | 0.000000 | 8.407791e-45 |
| int -3 | nan | nan |
| float 0x8A8B8C8D | 2324401408.000000 | 2.324401e+09 |
| float 6 | 6.000000 | 6.000000e+00 |
| float -3 | -3.000000 | -3.000000e+00 |

Код программы

#include <iostream>

#include <cstdlib>

using namespace std;

template<typename T>

void print64(T \*p)

{

cout << "Value: " << \*p <<endl;

cout.setf( ios::fixed );

cout <<"fixed: " << \*(reinterpret\_cast<double\*>(p)) << endl;

cout.unsetf( ios::fixed );

cout.setf( ios::scientific );

cout <<"scientific: "<< \*(reinterpret\_cast<double\*>(p)) << endl << endl;

cout.unsetf( ios::scientific );

}

template<typename T>

void print32(T \*p)

{

cout << "Value: " << \*p <<endl;

cout.setf( ios::fixed );

cout <<"fixed: " << \*(reinterpret\_cast<float\*>(p)) << endl;

cout.unsetf( ios::fixed );

cout.setf( ios::scientific );

cout <<"scientific: "<< \*(reinterpret\_cast<float\*>(p)) << endl << endl;

cout.unsetf( ios::scientific );

}

int main()

{

cout << "in 64 interpretation"<<endl;

unsigned long long umin = 0;//минимальное целое значение без знака 64 бит

unsigned long long umax = umin - 1;//максимальное целое значение без знака 64 бит

long long smin = -9223372036854775808;//мин. целое значение со знаком 64 бит

long long smax = 9223372036854775807;//макс. целое значение со знаком 64 бит

long long x = 0x8A8B8C8D; //значение x, соответствующее варианту

long long y = 6; //значение y, соответствующее варианту

long long z = -3; //значение z, соответствующее варианту

double X = 0x8A8B8C8D; //значение x, соответствующее варианту

double Y = 6; //значение y, соответствующее варианту

double Z = -3; //значение z, соответствующее варианту

print64(&umin);

print64(&umax);

print64(&smin);

print64(&smax);

print64(&x);

print64(&y);

print64(&z);

print64(&X);

print64(&Y);

print64(&Z);

cout << "in 32 interpretation"<<endl;

unsigned int min32 = 0;//Минимальное целое 32-битное значение без знака

unsigned int max32 =4294967295; //Макс. целое 32-битное значение без знака

int min\_sign = -2147483648;//Минимальное целое 32-битное значение со знаком

int max\_sign = 2147483647;//Максимальное целое 32-битное значение со знаком

int x\_i = 0x8A8B8C8D;

int y\_i = 6;

int z\_i = -3;

float x\_f = 0x8A8B8C8D;

float y\_f = 6;

float z\_f = -3;

print32(&min32);

print32(&max32);

print32(&min\_sign);

print32(&max\_sign);

print32(&x\_i);

print32(&y\_i);

print32(&z\_i);

print32(&x\_f);

print32(&y\_f);

print32(&z\_f);

return 0;

}

Для типа double

| **Значения** | **ж) интерпретация с плавающей запятой в представлении с фиксированным количеством цифр после запятой** | **з) интерпретация с плавающей запятой в экспоненциальном представлении** |
| --- | --- | --- |
| **0** | 0.000000 | 0.000000e+00 |
| **18446744073709551615** | nan | nan |
| **-9223372036854775808** | -0.000000 | -0.000000e+00 |
| **9223372036854775807** | nan | nan |
| **0x8A8B8C8D** | 0.000000 | 1.148407e-314 |
| **6** | 0.000000 | 2.964394e-323 |
| **-3** | nan | nan |
| **0x8A8B8C8D - double** | 2324401293.000000 | 2.324401e+09 |
| **6 - double** | 6.000000 | 6.000000e+00 |
| **-3 - double** | -3.000000 | -3.000000e+00 |

**Задание Л1.з5.**

Изучите, как располагаются в памяти байты, составляющие целое число и число с плавающей запятой. Для этого на языке C/C++ разработайте функцию void 𝑝𝑟𝑖𝑛𝑡𝐷𝑢𝑚𝑝(void \* 𝑝, size\_t 𝑁), которая печатает для области памяти по заданному адресу 𝑝 значения 𝑁 байтов, начиная с младшего, в шестнадцатеричном представлении (шестнадцатеричный дамп памяти). С помощью 𝑝𝑟𝑖𝑛𝑡𝐷𝑢𝑚𝑝() определите и выпишите в отчёт, как хранятся в памяти компьютера в программе на C/C++:

– целое число 𝑥 (типа 𝑖𝑛𝑡; таблица Л1.1); по результату исследования определите порядок следования байтов в словах для вашего процессора: 𝑥 =0x8A8B8C8D

**а) прямой (младший байт по младшему адресу, порядок Intel, Little-Endian, от младшего к старшему);**

б) обратный (младший байт по старшему адресу, порядок Motorola, BigEndian, от старшего к младшему);

– массив из трёх целых чисел (статический или динамический, но не высокоуровневый контейнер) с элементами 𝑥, 𝑦, 𝑧;

Dump Для х: 8D8C8B8A

Dump Для у: 06000000

Dump Для z: FDFFFFFF

Dump Для массива: 8D8C8B8A06000000FDFFFFFF

Можно сделать вывод, что **массив хранится в памяти компьютера как последовательность дампов его элементов в том же порядке, в котором они расположены в массиве**

– число с плавающей запятой 𝑦 (типа 𝑑𝑜𝑢𝑏𝑙𝑒; таблица Л1.1). Это число 6

Dump:0000000000001840

6 = 1102

Сдвинем число на 2 разрядов вправо. В результате мы получили основные составляющие экспоненциального нормализованного двоичного числа:

Мантисса M=1.100

Экспонента exp2=2

**Преобразование двоичного нормализованного числа в 64 битный формат IEEE 754**.

Первый бит отводится для обозначения знака числа. Поскольку число положительное, то первый бит равен 0

Следующие 11 бит (с 2-го по 12-й) отведены под экспоненту.

Для определения знака экспоненты, чтобы не вводить ещё один бит знака, добавляют смещение к экспоненте +1023. Таким образом, наша экспонента: 2 + 1023 = 1025

Переведем экспоненту в двоичное представление.

1025 = 100000000012

Оставшиеся 52 бита отводят для мантиссы. У нормализованной двоичной мантиссы первый бит всегда равен 1, так как число лежит в диапазоне 1 ≤ M < 2. Для экономии, единицу не записывают, а записывают только остаток от мантиссы: 1000000000000000000000000000000000000000000000000000

В результате число 6 представленное в IEEE 754 c двойной (double) точностью равно 0100000000011000000000000000000000000000000000000000000000000000.

Переведем в шестнадцатеричное представление.

Разделим исходный код на группы по 4 разряда.

01000000000110000000000000000000000000000000000000000000000000002 = 0100 0000 0001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 2

Получаем число: (один байт обозначен жирным, всего 8 байт)

**0100 0000** 0001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 2 = 401800000000000016

В нашем случае байты записываются в память от младшего к старшему, и конечный результат будет иметь вид - 000000000000184016

Код программы:

#include <iostream>

#include <iomanip>

using namespace std;

template<typename T>

void printDump(const T &x, int N){

const unsigned char \*p = reinterpret\_cast<const unsigned char \*>(&x);

cout<<"Value = "<<x<<endl<<"Dump:"<<hex<<uppercase<<setfill('0');

for(size\_t i = 0; i < N; ++i)

{

cout << setw(2)<< static\_cast<unsigned>(\*(p+i));

}

cout << dec << endl << endl;

}

int main()

{

int x = 0x8A8B8C8D;

int y = 6;

int z = -3;

int mas[3] = {x,y,z};

double d = 6;

printDump(x, 4);

printDump(y, 4);

printDump(z, 4);

printDump(mas, 12);

printDump(d, 8);

return 0;

}

