**Лабораторная работа 2**

**Представление данных в ЭВМ. Сравнение различных платформ.**

**Задание Л2.з1.** Выполните измерения согласно заданию Л1.з2 на платформах, доступных на ВЦ (таблица Л2.1). Результаты оформите в отчёте в виде таблицы.

Примечание: связка GNU/Linux 64 + GCC 64 широко используется в онлайн-компиляторах. На godbolt.org (ОС GNU/Linux 64) доступны сборка компиляторами GCC 64, clang 64 и ICC (Intel C++ Compiler) 64 с возможностью запуска; а также сборка без запуска для множества других компиляторов, в том числе для не-x86 процессоров. ОС MS Windows 64 и компиляторы GCC и Microsoft доступны на ВЦ локально (для дистанционных занятий — на терминале ВЦ). Не возбраняется использование установленных дома инструментов таблицы Л2.1

**Штраф** −2 балла за платформу таблицы Л2.1, если в аудитории она доступна, а данных по ней нет.

**Бонус** +2 балла за платформу. При подготовке к работе выполните измерения на платформе, отсутствующей в таблице Л2.1 (укажите ОС, компилятор, режим (разрядность) сборки, архитектуру процессора, назначение платформы — без этих сведений баллы не начисляются).

|  | **GNU/Linux 64** | | | **MS Windows 64** | | | **Mac OS 64** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GCC 64** | **clang 64** | **Intel 64** | **GCC (MinGW) 64** | **Microsoft 64** | **Microsoft 32** | **Apple clang** |
| **char** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **signed char** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **unsigned char** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **wchar\_t** | 4 | 4 | 4 | 2 | 2 | 2 | 4 |
| **short** | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **unsigned short** | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **int** | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **unsigned int** | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **long** | 8 | 8 | 8 | 4 | 4 | 4 | 8 |
| **unsigned long** | 8 | 8 | 8 | 4 | 4 | 4 | 8 |
| **long long** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| **unsigned long long** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| **float** | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **double** | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| **long double** | 16 | 16 | 16 | 16 | 8 | 8 | 16 |
| **size\_t** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |
| **ptrdiff\_t** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |
| **void\*** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |
| **char\*** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |
| **int\*** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |
| **unsigned int\*** | 8 | 8 | 8 | 8 | 8 | 4 | 8 |

| **Microsoft 32** | **Microsoft 64** | **GCC (MinGW) 64** |
| --- | --- | --- |
|  |  |  |

| **GNU/Linux 64 GCC 64** | **GNU/Linux 64 clang 64** | **GNU/Linux 64 Intel 64** |
| --- | --- | --- |
|  |  |  |

**Код программы**

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}

**Задание Л2.з2.** Выполните на платформах из таблицы Л2.1 задание Л1.з5. Убедитесь, что на всех платформах на основе x86 и x86-64 порядок байтов прямой. Бонус +2 балла за платформу. При подготовке к работе выполните измерения на платформе, где архитектура процессора отлична от x86/x86-64.

| **OС** | **Компилятор, разрядность сборки** | **Порядок байтов** | **Целое число** | **Массив из трёх целых чисел** | **Число с плавающей запятой** |
| --- | --- | --- | --- | --- | --- |
| **GNU/Linux 64** | **GCC 64** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **GNU/Linux 64** | **clang 64** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **GNU/Linux 64** | **Intel 64** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **MS Windows 64** | **GCC (MinGW) 64** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **MS Windows 64** | **Microsoft 64** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **MS Windows 64** | **Microsoft 32** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |
| **Mac OS 64** | **Apple clang** | прямой | 8D8C8B8A | 8D8C8B8A06000000FDFFFFFF | 0000000000001840 |

**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;

}

**Задание Л2.з3.** С помощью функции 𝑝𝑟𝑖𝑛𝑡𝐷𝑢𝑚𝑝() задания Л1.з5 определите и выпишите в отчёт, как хранятся в памяти на платформах из таблицы Л2.1:

– строки "abczklmn" и "абвёклмн" из char; при выборе количества отображаемых байтов 𝑁 учитывайте всю длину строки (включая завершающий нулевой символ), а не только видимые буквы;

– «широкие» строки L"abczklmn" и L"абвёклмн" из wchar\_t; при выборе 𝑁 учитывайте всю длину строки. Результаты оформите в отчёте в виде таблицы.

Примечание: на MS Windows возможна (если файл исходного кода сохранён в однобайтовой кодировке windows-1251) ситуация, когда литерал L"абвёклмн" не воспринимается компилятором как корректная широкая строка. Поставьте в соответствующих ячейках отчёта прочерки.

Дополнительные бонусные и штрафные баллы −3 балла за утечку памяти (то есть наличие выделенных, но не освобождённых блоков динамической памяти).

| **OС** | **Компилятор, разрядность сборки** | **abczklmn char** | **абвёклмн char** | **L"abczklmn" wchar\_t** | **L"абвёклмн" wchar\_t** |
| --- | --- | --- | --- | --- | --- |
| **GNU/Linux 64** | **GCC 64** | 61 62 63 7A 6B 6C 6D 6E 00 | D0 B0 D0 B1 D0 B2 D1 91 D0 BA D0 BB D0 BC D0 BD 00 | 6100 0000 6200 0000 6300 0000 7A00 0000 6B00 0000 6C00 0000 6D00 0000 6E00 0000 0000 0000 | 3004 0000 3104 0000 3204 0000 5104 0000 3A04 0000 3B04 0000 3C04 0000 3D04 0000 0000 0000 |
| **GNU/Linux 64** | **clang 64** | 61 62 63 7A 6B 6C 6D 6E 00 | D0 B0 D0 B1 D0 B2 D1 91 D0 BA D0 BB D0 BC D0 BD 00 | 6100 0000 6200 0000 6300 0000 7A00 0000 6B00 0000 6C00 0000 6D00 0000 6E00 0000 0000 0000 | 3004 0000 3104 0000 3204 0000 5104 0000 3A04 0000 3B04 0000 3C04 0000 3D04 0000 0000 0000 |
| **GNU/Linux 64** | **Intel 64** | 61 62 63 7A 6B 6C 6D 6E 00 | D0 B0 D0 B1 D0 B2 D1 91 D0 BA D0 BB D0 BC D0 BD 00 | 61000000 62000000 63000000 7A000000 6B000000 6C000000 6D000000 6E000000 00000000 | 30040000 31040000 32040000 51040000 3A040000 3B040000 3C040000 3D040000 00000000 |
| **MS Windows 64** | **GCC (MinGW) 64** | 61 62 63 7A 6B 6C 6D 6E 00 | E0 E1 E2 B8 EA EB EC ED 00 | 6100 6200 6300 7A00 6B00 6C00 6D00 6E00 0000 | 3004 3104 3204 5104 3A04 3B04 3C04 3D04 0000 |
| **MS Windows 64** | **Microsoft 64** | 61 62 63 7A 6B 6C 6D 6E 00 | E0 E1 E2 B8 EA EB EC ED 00 | 6100 6200 6300 7A00 6B00 6C00 6D00 6E00 0000 | 3004 3104 3204 5104 3A04 3B04 3C04 3D04 0000 |
| **MS Windows 64** | **Microsoft 32** | 61 62 63 7A 6B 6C 6D 6E 00 | E0 E1 E2 B8 EA EB EC ED 00 | 6100 6200 6300 7A00 6B00 6C00 6D00 6E00 0000 | 3004 3104 3204 5104 3A04 3B04 3C04 3D04 0000 |
| **Mac OS 64** | **Apple clang** | 61 62 63 7A 6B 6C 6D 6E 00 | D0 B0 D0 B1 D0 B2 D1 91 D0 BA D0 BB D0 BC D0 BD 00 | 6100 0000 6200 0000 6300 0000 7A00 0000 6B00 0000 6C00 0000 6D00 0000 6E00 0000 0000 0000 | 3004 0000 3104 0000 3204 0000 5104 0000 3A04 0000 3B04 0000 3C04 0000 3D04 0000 0000 0000 |

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNING

#include <stdio.h>

#include <iostream>

#include <iomanip>

#include <Windows.h>

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()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char str1[] = "abczklmn";

char str2[] = "абвёклмн";

wchar\_t str3[] = L"abczklmn";

wchar\_t str4[] = L"абвёклмн";

printDump(str1, sizeof((str1));

printDump(str2, sizeof(str2));

printDump(str3, sizeof(str3));

printDump(str4, sizeof(str4));

return 0;

}