Предисловие

Кодировка

Язык моей системы – английский, поэтому способы вроде:

setlocale (LC\_ALL," Russian ");

Не помогут использовать русский текст в консоли.

Поэтому я использовал способ который будет работать на любом компьютере, а точнее использование 2-х байтовых символов в кодировке Unicode.

Для этого надо:

1. В начале программы необходимо настроить потоки ввода, вывода и вывода ошибок:

if (\_setmode(\_fileno(stdout), \_O\_U16TEXT) == -1) return -1;

if (\_setmode(\_fileno(stdin ), \_O\_U16TEXT) == -1) return -1;

if (\_setmode(\_fileno(stderr), \_O\_U16TEXT) == -1) return -1;

Для этого могут понадобится такие заголовки:

#include <io.h>

#include <fcntl.h>

1. Вместо функций printf и scanf\_s необходимо использовать функции wprintf и wscanf\_s соответственно;
2. Вместо типа char необходимо использовать тип wchar\_t;
3. Перед литералами строк необходимо ставить букву L.

Общий шаблон всех программ

#include <iostream>

#include <io.h>

#include <fcntl.h>

//ToDo Другие заголовки

int main()

{

if (\_setmode(\_fileno(stdout), \_O\_U16TEXT) == -1) return -1;

if (\_setmode(\_fileno(stdin ), \_O\_U16TEXT) == -1) return -1;

if (\_setmode(\_fileno(stderr), \_O\_U16TEXT) == -1) return -1;

//ToDo Тело программы

}

ЛР4: Побитовые операции

Задание

Формат команды канала в система ввода-вывода имеет вид:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **╧ разряда** | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| **Значение** | O | O | O | O | O | C | D | N | N | N | N | N | N | N | N | N |

Где:

* O..O - код операции
* C - признак цепочки команд
* D - признак цепочки данных
* N..N - количество байтов для передачи

Комментарии к решению

Чтоб избежать дублирования кода – я добавил 2 функции:

inline void WriteBits(uint16\_t& n, int bits, int from, int count)

{

auto mask = (1 << count) - 1;

n |= (bits & mask) << from;

}

inline int ReadBits(uint16\_t source, int from, int count)

{

auto mask = (1 << count) - 1;

return (source >> from) & mask;

}

Переменная mask является числом, у которого только count нижних бит заполнено единицами, а все остальные биты – нули.

Тип первого агрумента WriteBits - uint16\_t&, где & значит передачу по ссылке.

Так же, я добавил макро RE\_READ\_ENCODED в начало программы, чтоб можно было быстро включать/выключать необходимость перезаписи зашифрованного значения с клавиатуры.

Весь код

#include <iostream>

#include <io.h>

#include <fcntl.h>

#define RE\_READ\_ENCODED true

/// <summary>

/// Записывает count битов из числа bits в разряд from числа n

/// </summary>

inline void WriteBits(uint16\_t& n, int bits, int from, int count)

{

auto mask = (1 << count) - 1;

n |= (bits & mask) << from;

}

/// <summary>

/// Вырезает и возвращается область из count бит числа source, начиная с разряда from

/// </summary>

inline int ReadBits(uint16\_t source, int from, int count)

{

auto mask = (1 << count) - 1;

return (source >> from) & mask;

}

int main()

{

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

\_setmode(\_fileno(stdin ), \_O\_U16TEXT);

\_setmode(\_fileno(stderr), \_O\_U16TEXT);

int oper\_code, command\_chain, data\_chain, byte\_count;

wprintf(L"Код операции (0..31): ");

wscanf\_s(L"%d", &oper\_code);

wprintf(L"Признак цепочки команд (0..1): ");

wscanf\_s(L"%d", &command\_chain);

command\_chain = command\_chain ? 1 : 0;

wprintf(L"Признак цепочки данных (0..1): ");

wscanf\_s(L"%d", &data\_chain);

data\_chain = data\_chain ? 1 : 0;

wprintf(L"Количество байтов для передачи (0..511): ");

wscanf\_s(L"%d", &byte\_count);

uint16\_t encoded{};

WriteBits(encoded, byte\_count, 0, 9);

WriteBits(encoded, data\_chain, 9, 1);

WriteBits(encoded, command\_chain, 10, 1);

WriteBits(encoded, oper\_code, 11, 5);

wprintf(L"В зашифрованном виде: 0x%X\n", encoded);

if (RE\_READ\_ENCODED)

{

wprintf(L"Введите зашифрованное состояние в 16СС: ");

int temp\_encoded;

wscanf\_s(L"%x", &temp\_encoded);

encoded = temp\_encoded;

}

wprintf(L"Код операции: %d\n", ReadBits(encoded, 11, 5));

wprintf(L"Признак цепочки команд: %d\n", ReadBits(encoded, 10, 1));

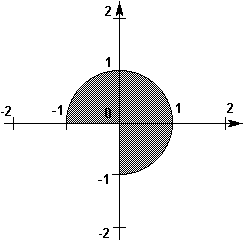
wprintf(L"Признак цепочки данных: %d\n", ReadBits(encoded, 9, 1));

wprintf(L"Количество байтов для передачи: %d\n", ReadBits(encoded, 0, 9));

}

ЛР 5: Условный оператор

Задание



Комментарии к решению

Сначала я проверяю попадает в круг с радиусом 1 и центром в (0; 0), затем – не\_принадлежность к III четверти.

Вообще, ещё можно было бы написать так:

auto res\_descr = условие ? "попадает" : "НЕ попадает";

И при выводе использовать эту строку, чтоб минимизировать дубли кода.

Весь код

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main()

{

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

\_setmode(\_fileno(stdin ), \_O\_U16TEXT);

\_setmode(\_fileno(stderr), \_O\_U16TEXT);

wprintf(L"Введите [x] и [y] через пробел: ");

double x, y;

wscanf\_s(L"%lf %lf", &x, &y);

if (x \* x + y \* y < 1 \* 1 && (x > 0 || y > 0))

wprintf(L"Точка (x=%lf; y=%lf) попадает в область", x, y); else

wprintf(L"Точка (x=%lf; y=%lf) НЕ попадает в область", x, y);

}

ЛР 6: Операторы цикла

Задание



Комментарии к коду

Для вычисления суммы первых 10 элементов я использовал цикл for.

Для вычисления суммы элементов >e так же можно было использовать цикл for, вот так:

for (double last\_item, int i = 0; abs(last\_item = item\_at(i)) > e; ++i)

Это, кроме всего прочего, немного эффективнее, потому что ++i считается после тела цикла. Но так выглядит более громоздко, поэтому я использовал цикл while и расписал всё подробнее.

Так же, я добавил дополнительные вложенные блоки кода { }, чтоб отделить области видимости переменных. Таким образом я могу объявить 2 разные переменные sum в одной подпрограмме

Весь код

#include <iostream>

#include <io.h>

#include <fcntl.h>

const double e = 0.000001;

/// <summary>

/// Вычисляет элемент ряда в номер n (считая с 0)

/// </summary>

double item\_at(int n)

{

return

((n % 2) ? -1 : 1) \*

((double)n + 1) /

pow(2, n - 1);

}

int main()

{

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

\_setmode(\_fileno(stdin ), \_O\_U16TEXT);

\_setmode(\_fileno(stderr), \_O\_U16TEXT);

{

auto sum = 0.0;

for (auto i = 0; i < 10; ++i)

sum += item\_at(i);

wprintf(L"Сумма первых 10 элементов равна %lf\n", sum);

}

{

auto sum = 0.0;

double last\_item;

auto i = 0;

while (abs(last\_item = item\_at(i++)) > e)

sum += last\_item;

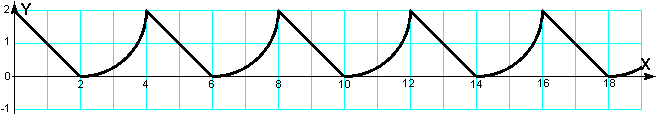
wprintf(L"Сумма первых %d элементов, каждый из которых >%e, равна %lf\n", i, e, sum);

}

}

ЛР 7: Операторы цикла

Задание



Комментарии к коду

Весь код

ЛР 8: Работа с массивами

Задание

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Размерность массива** | **Диапазон значений** | **Что нужно сделать** |
| **9** | 200 | 0 - 100 | Найти непрерывный участок из 10 элементов, сумма которых максимальна |

Комментарии к коду

Весь код

ЛР 9: Работа с матрицами

Задание

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Содержание задания** | **Иллюстрация** |
| **9** | Заполнить матрицу ЛП, от левого нижнего угла по диагонали: влево - вверх. |  |

Комментарии к коду

Весь код