Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

**ПРАКТИЧЕСКАЯ РАБОТА № 13**

**Тема:** Ввод-вывод на консоль

**Специальность: 09.02.07 Информационные системы и программирование,**

**по квалификации «Программист»**

Группа ИП-21-7К\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент БобриковАА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф. И. О

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Санкт-Петербург

2023 г

**Листинг 13.1. Чтение и запись на консоль посредством функций ReadConsole и WriteConsole**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

HANDLE hStdOut, hStdIn; // дескрипторы консоли

DWORD dwWritten, dwRead; // для количества символов

char buffer[80]; // для ввода символов

char str[] = "Input any string:";

char c;

// читаем дескрипторы консоли

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

hStdIn = GetStdHandle(STD\_INPUT\_HANDLE);

if (hStdOut == INVALID\_HANDLE\_VALUE || hStdIn == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard handle failed." << endl;

return GetLastError();

}

// выводим сообщения о вводе строки

if (!WriteConsole(hStdOut, &str, sizeof(str), &dwWritten, NULL))

{

cout << "Write console failed." << endl;

return GetLastError();

}

// вводим строку

if (!ReadConsole(hStdIn, &buffer, sizeof(buffer), &dwRead, NULL))

{

cout << "Read console failed." << endl;

return GetLastError();

}

// ждем команду на завершение работы

cout << "Input any char to exit: ";

cin >> c;

return 0;

}



**Листинг 13.2. Чтение и запись на консоль посредством функций ReadFile и WriteFile**

**Листинг 13.3. Чтение записей входного буфера консоли**

#include <windows.h>

#include <iostream>

using namespace std;

HANDLE hStdIn, hStdOut; // для дескрипторов стандартного ввода и вывода

BOOL bRead = TRUE; // для цикла обработки событий

// функция обработки сообщений от клавиатуры

VOID KeyEventProc(KEY\_EVENT\_RECORD kir)

{

cout << "\tKey event record:" << endl;

// просто выводим на консоль содержимое записи

cout << "bKeyDown = " << hex << kir.bKeyDown << endl;

cout << "wRepeatCount = " << dec << kir.wRepeatCount << endl;

cout << "wVirtualKeyCode = " << hex << kir.wVirtualKeyCode << endl;

cout << "wVirtualScanCode = " << kir.wVirtualScanCode << endl;

cout << "uChar.AsciiChar = " << kir.uChar.AsciiChar << endl;

cout << "dwControlKeyState = " << kir.dwControlKeyState << endl;

// если ввели букву 'q', то выходим из цикла обработки событий

if (kir.uChar.AsciiChar == 'q')

bRead = FALSE;

}

// функция обработки сообщений от мыши

VOID MouseEventProc(MOUSE\_EVENT\_RECORD mer)

{

cout << "\tMouse event record:" << endl << dec;

// просто выводим на консоль содержимое записи

cout << "dwMousePosition.X = " << mer.dwMousePosition.X << endl;

cout << "dwMousePosition.Y = " << mer.dwMousePosition.Y << endl;

cout << "dwButtonState = " << hex << mer.dwButtonState << endl;

cout << "dwControlKeyState = " << mer.dwControlKeyState << endl;

cout << "dwEventFlags = " << mer.dwEventFlags << endl;

}

// функция обработки сообщения об изменении размеров окна

VOID ResizeEventProc(WINDOW\_BUFFER\_SIZE\_RECORD wbsr)

{

// изменяем размеры буфера вывода

SetConsoleScreenBufferSize(hStdOut, wbsr.dwSize);

}

int main()

{

INPUT\_RECORD ir; // входная запись

DWORD cNumRead; // для количества прочитанных записей

// получить дескрипторы стандартного ввода и вывода

hStdIn = GetStdHandle(STD\_INPUT\_HANDLE);

if (hStdIn == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard input handle failed." << endl;

return GetLastError();

}

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (hStdOut == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard output handle failed." << endl;

return GetLastError();

}

// начинаем обработку событий ввода

cout << "Begin input event queue processing." << endl;

cout << "Input 'q' to quit." << endl << endl;

// цикл обработки событий ввода

while (bRead)

{

// ждем событие ввода

WaitForSingleObject(hStdIn, INFINITE);

// читаем запись ввода

if (!ReadConsoleInput(

hStdIn, // дескриптор ввода

&ir, // буфер для записи

1, // читаем одну запись

&cNumRead)) // количество прочитанных записей

{

cout << "Read console input failed." << endl;

break;

}

// вызываем соответствующий обработчик

switch (ir.EventType)

{

case KEY\_EVENT: // событие ввода с клавиатуры

KeyEventProc(ir.Event.KeyEvent);

break;

case MOUSE\_EVENT: // событие ввода с мыши

MouseEventProc(ir.Event.MouseEvent);

break;

case WINDOW\_BUFFER\_SIZE\_EVENT: // изменения размеров окна

ResizeEventProc(

ir.Event.WindowBufferSizeEvent);

break;

case FOCUS\_EVENT: // события фокуса ввода игнорируем

break;

case MENU\_EVENT: // события меню игнорируем

break;

default: // неизвестное событие

cout << "Unknown event type.";

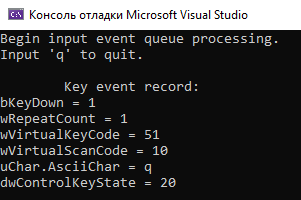
break;

}

}

return 0;

}



**Листинг 13.4. Пример записи в буфер ввода, подсчет количества записей и очистка буфера ввода**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

HANDLE hStdIn; // для дескриптора стандартного ввода

INPUT\_RECORD ir; // входная запись

DWORD dwNumberWritten; // количество записанных записей

DWORD dwNumber; // для количества записей в буфере ввода

// получить дескриптор стандартного ввода

hStdIn = GetStdHandle(STD\_INPUT\_HANDLE);

if (hStdIn == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard input handle failed." << endl;

return GetLastError();

}

// подсчитываем записи в буфере ввода

if (!GetNumberOfConsoleInputEvents(hStdIn, &dwNumber))

{

cout << "Get number of console input events failed." << endl;

return GetLastError();

}

// печатаем количество событий ввода

cout << "Number of console input events = " << dwNumber << endl;

// инициализируем запись события ввода

ir.EventType = KEY\_EVENT;

ir.Event.KeyEvent.bKeyDown = 0x1;

ir.Event.KeyEvent.wRepeatCount = 1;

ir.Event.KeyEvent.wVirtualKeyCode = 0x43;

ir.Event.KeyEvent.wVirtualScanCode = 0x2e;

ir.Event.KeyEvent.uChar.AsciiChar = 'c';

ir.Event.KeyEvent.dwControlKeyState = 0x20;

// записываем запись в буфер ввода

if (!WriteConsoleInput(hStdIn, &ir, 1, &dwNumberWritten))

{

cout << "Write console input failed." << endl;

return GetLastError();

}

cout << "Write one record into the input buffer." << endl;

// подсчитываем записи в буфере ввода

if (!GetNumberOfConsoleInputEvents(hStdIn, &dwNumber))

{

cout << "Get number of console input events failed." << endl;

return GetLastError();

}

// печатаем количество событий ввода

cout << "Number of console input events = " << dwNumber << endl;

// очищаем входной буфер

cout << "Flush console input buffer." << endl;

if (!FlushConsoleInputBuffer(hStdIn))

{

cout << "Flush console input buffer failed." << endl;

return GetLastError();

}

// подсчитываем записи в буфере ввода

if (!GetNumberOfConsoleInputEvents(hStdIn, &dwNumber))

{

cout << "Get number of console input events failed." << endl;

return GetLastError();

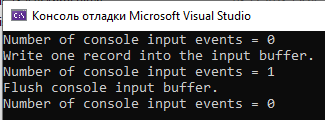
}

// печатаем количество событий ввода

cout << "Number of console input events = " << dwNumber << endl;

return 0;

}



**Листинг 13.5. Определение количества кнопок у мыши**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

DWORD dwNumber; // для количества кнопок у мыши

// подсчитываем количество кнопок у мыши

if (!GetNumberOfConsoleMouseButtons(&dwNumber))

{

cout << "Get number of console mouse buttons failed." << endl;

return GetLastError();

}

// выводим количество кнопок у мыши

cout << "Number of console mouse buttons = " << dwNumber << endl;

return 0;

}



**Листинг 13.6. Чтение последовательности символов из буфера экрана**

**Листинг 13.7. Запись последовательности символов в буфер экрана**

**Листинг 13.8. Заполнение буфера экрана заданным символом**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

char c;

HANDLE hStdOut; // дескриптор стандартного вывода

DWORD dwLength; // количество заполняемых клеток

DWORD dwWritten; // для количества заполенных клеток

COORD coord; // координаты первой клетки

CONSOLE\_SCREEN\_BUFFER\_INFO csbi; // для параметров буфера экрана

// читаем дескриптор стандартного вывода

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (hStdOut == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard handle failed." << endl;

return GetLastError();

}

// читаем параметры выходного буфера

if (!GetConsoleScreenBufferInfo(hStdOut, &csbi))

{

cout << "Console screen buffer info failed." << endl;

return GetLastError();

}

// вычисляем размер буфера экрана в символах

dwLength = csbi.dwSize.X \* csbi.dwSize.Y;

// устанавливаем координаты первой клетки

coord.X = 0;

coord.Y = 0;

// вводим символ-заполнитель

cout << "Input any char to fill screen buffer: ";

cin >> c;

// заполняем буфер экрана символом-заполнителем

if (!FillConsoleOutputCharacter(

hStdOut, // стандартный дескриптор вывода

c, // символ заполнения

dwLength, // длина буфера в символах

coord, // индекс первой клетки

&dwWritten)) // количество заполненных клеток

{

cout << "Fill console output character failed." << endl;

return GetLastError();

}

// ждем команды на очищение буфера экрана

cout << "In order to clear screen buffer, press any char: ";

cin >> c;

// очищаем буфер экрана пробелами

if (!FillConsoleOutputCharacter(

hStdOut, // стандартный дескриптор вывода

' ', // символ заполнения

dwLength, // длина буфера в символах

coord, // индекс первой клетки

& dwWritten)) // количество заполненных клеток

{

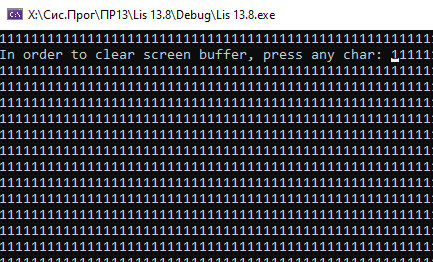
cout << "Fill console output character failed." << endl;

return GetLastError();

}

return 0;

}



**Листинг 13.9. Чтение прямоугольной области из буфера экрана**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

HANDLE hConsoleOutput; // для дескриптора буфера экрана

CHAR\_INFO lpBuffer[4]; // буфер для ввода

COORD dwBufferSize = { 2, 2 }; // размеры буфера

COORD dwBufferCoord = { 0, 0 }; // координаты первого элемента в буфере

SMALL\_RECT ReadRegion = { 0, 0, 1, 1 }; // прямоугольник, который читаем

// выводим символы, которые будем читать

cout << 'a' << 'b' << endl << 'c' << 'd' << endl;

// получаем дескриптор ввода

hConsoleOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (hConsoleOutput == INVALID\_HANDLE\_VALUE)

{

cout << "Get standard handle failed." << endl;

return GetLastError();

}

// читаем символы

if (!ReadConsoleOutput(hConsoleOutput, lpBuffer, dwBufferSize,

dwBufferCoord, &ReadRegion))

{

cout << "Read console input failed." << endl;

return GetLastError();

}

// распечатываем прочитанные символы

cout << "Read cells." << hex << endl;

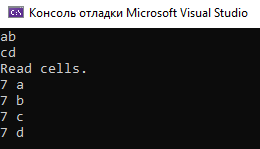
for (int i = 0; i < 4; ++i)

cout << lpBuffer[i].Attributes << ' ' << lpBuffer[i].Char.AsciiChar

<< endl;

return 0;

}



**Листинг 13.10. Заполнение прямоугольной области в буфере экрана**

#include <windows.h>

#include <iostream>

using namespace std;

int main()

{

HANDLE hStdOut; // дескриптор стандартного вывода

CHAR\_INFO ci[80 \* 25]; // прямоугольник, из которого будем выводить

COORD size; // размеры этого прямоугольника

// координаты левого угла прямоугольника, из которого выводим

COORD coord;

// координаты левого угла прямоугольника, в который пишем

SMALL\_RECT sr;

// читаем стандартный дескриптор вывода

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

// заполняем прямоугольник, который будем выводить, пробелами

for (int i = 0; i < 80 \* 25; ++i)

{

ci[i].Char.AsciiChar = ' ';

ci[i].Attributes = BACKGROUND\_BLUE | BACKGROUND\_INTENSITY;

}

// устанавливаем левый угол многоугольника, из которого пишем

coord.X = 0;

coord.Y = 0;

// устанавливаем размеры прямоугольника, который пишем

size.X = 80;

size.Y = 25;

// вводим координаты левого верхнего угла многоугольника,

// в который пишем

cout << "Input left coordinate to write: ";

cin >> sr.Left;

cout << "Input top coordinate to write: ";

cin >> sr.Top;

// вводим координаты правого нижнего угла прямоугольника,

// в который пишем

cout << "Input right coordinate to write: ";

cin >> sr.Right;

cout << "Input down coordinate to write: ";

cin >> sr.Bottom;

// пишем прямоугольник в буфер экрана

if (!WriteConsoleOutput(

hStdOut, // дескриптор буфера экрана

ci, // прямоугольник, из которого пишем

size, // размеры этого прямоугольника

coord, // и его левый угол

&sr)) // прямоугольник, в который пишем

{

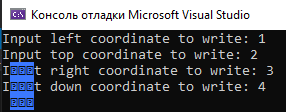
cout << "Write console output failed." << endl;

return GetLastError();

}

return 0;

}



**Листинг 13.11. Отключение режима эхо-вывода**

**Листинг 13.12. Прокрутка буфера экрана**