Лабораторная работа: Работа с буфером экрана консоли

Цель

Научиться создавать и использовать буферы экрана, а также управлять их параметрами и отображением текста.

1. Создание и активация буфера экрана

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

// Выводим сообщение в новом буфере

DWORD charsWritten;

WriteConsoleW(hScreenBuffer, L"Буфер экрана успешно создан и активирован!\n", wcslen(L"Буфер экрана успешно создан и активирован!\n"), &charsWritten, NULL);

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

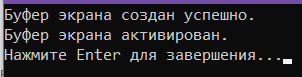
std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Результат:



2. Установка параметров буфера (не обязательно к выполнению)

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

COORD bufferSize = { 120, 40 }; // Задаём ширину и высоту буфера

if (!SetConsoleScreenBufferSize(hScreenBuffer, bufferSize)) {

std::wcout << L"Ошибка установки размера буфера экрана.\n";

}

else {

std::wcout << L"Размер буфера экрана установлен.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

// Выводим сообщение в новом буфере

DWORD charsWritten;

WriteConsoleW(hScreenBuffer, L"Буфер экрана успешно создан и активирован!\n", wcslen(L"Буфер экрана успешно создан и активирован!\n"), &charsWritten, NULL);

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

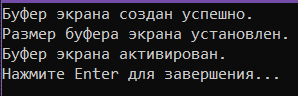
std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Результат:



3. Управление курсором

Изменим позицию курсора в буфере экрана.

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

COORD bufferSize = { 120, 40 }; // Задаём ширину и высоту буфера

if (!SetConsoleScreenBufferSize(hScreenBuffer, bufferSize)) {

std::wcout << L"Ошибка установки размера буфера экрана.\n";

}

else {

std::wcout << L"Размер буфера экрана установлен.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

// Выводим сообщение в новом буфере

DWORD charsWritten;

WriteConsoleW(hScreenBuffer, L"Буфер экрана успешно создан и активирован!\n", wcslen(L"Буфер экрана успешно создан и активирован!\n"), &charsWritten, NULL);

COORD cursorPos = { 5, 5 }; // Позиция курсора

if (!SetConsoleCursorPosition(hScreenBuffer, cursorPos)) {

std::wcout << L"Ошибка установки позиции курсора.\n";

}

else {

std::wcout << L"Курсор перемещён на позицию (5, 5).\n";

}

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

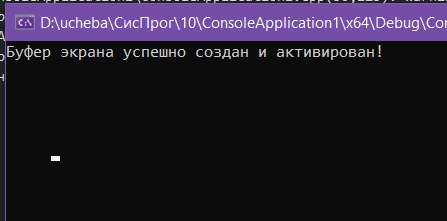
std::wcout << L"Нажмите Enter для завершения...";

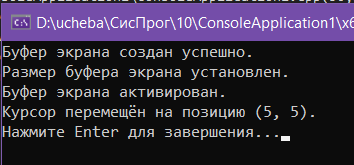
std::wcin.get();

return 0;

}

Результат:





4. Чтение и установка атрибутов

Изменим цвет текста и фона в буфере экрана.

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

COORD bufferSize = { 140, 90 }; // Задаём ширину и высоту буфера

if (!SetConsoleScreenBufferSize(hScreenBuffer, bufferSize)) {

std::wcout << L"Ошибка установки размера буфера экрана.\n";

}

else {

std::wcout << L"Размер буфера экрана установлен.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

SetConsoleTextAttribute(hScreenBuffer, FOREGROUND\_BLUE | BACKGROUND\_RED);

// Выводим сообщение в новом буфере

DWORD charsWritten;

WriteConsoleW(hScreenBuffer, L"Буфер экрана успешно создан и активирован! (Текст с синим текстом на красном фоне.)\n", wcslen(L"Буфер экрана успешно создан и активирован! (Текст с синим текстом на красном фоне.)\n"), &charsWritten, NULL);

;

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

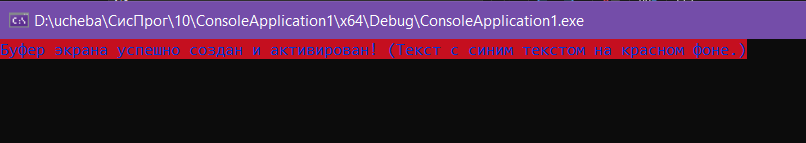
std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Результат:



Самостоятельные задания

1. Изменение размеров буфера и окна консоли. (не обязательно к выполнению)

Попробуйте изменить параметры ширины и высоты буфера и окна.

Подсказка: используйте SetConsoleScreenBufferSize для изменения размеров буфера и MoveWindow для изменения окна консоли.

Пример:

COORD newSize = { 100, 30 };

SetConsoleScreenBufferSize(hScreenBuffer, newSize);

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

// Устанавливаем размер буфера экрана

COORD newSize = { 80, 25 }; // Ширина = 80, Высота = 25

if (!SetConsoleScreenBufferSize(hScreenBuffer, newSize)) {

DWORD error = GetLastError();

std::wcout << L"Ошибка изменения размера буфера экрана. Код ошибки: " << error << L"\n";

CloseHandle(hScreenBuffer);

return 1;

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

// Изменяем размер окна консоли

HWND hConsole = GetConsoleWindow();

MoveWindow(hConsole, 100, 100, 800, 600, TRUE); // Позиция (100, 100), размер (800, 600)

// Выводим сообщение в новом буфере

DWORD charsWritten;

WriteConsoleW(hScreenBuffer, L"Буфер экрана успешно создан и активирован!\n", wcslen(L"Буфер экрана успешно создан и активирован!\n"), &charsWritten, NULL);

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Результат:



1. Добавление текста в созданный буфер экрана с изменёнными атрибутами.

Выведите текст в новом буфере с разными цветами.

Подсказка: используйте SetConsoleTextAttribute и WriteConsoleW для вывода текста в буфер.

Пример:

SetConsoleTextAttribute(hScreenBuffer, FOREGROUND\_GREEN | BACKGROUND\_BLUE);

WriteConsoleW(hScreenBuffer, L"Текст с зелёным текстом на синем фоне.\n", 40, &charsWritten, NULL);

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

// Выводим текст с разными цветами

DWORD charsWritten;

// Текст с зелёным текстом на синем фоне

SetConsoleTextAttribute(hScreenBuffer, FOREGROUND\_GREEN | BACKGROUND\_BLUE);

WriteConsoleW(hScreenBuffer, L"Текст с зелёным текстом на синем фоне.\n", wcslen(L"Текст с зелёным текстом на синем фоне.\n"), &charsWritten, NULL);

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

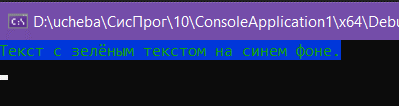
std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Код:



1. Изменение положения курсора и текста в разных местах экрана.

Установите курсор на разные позиции и выводите текст в этих местах.

Подсказка: Используйте SetConsoleCursorPosition и задайте несколько координат.

Пример:

COORD pos = { 10, 10 };

SetConsoleCursorPosition(hScreenBuffer, pos);

WriteConsoleW(hScreenBuffer, L"Текст на позиции 10, 10", 20, &charsWritten, NULL);

Код:

#include <windows.h>

#include <iostream>

#include <io.h>

#include <fcntl.h>

int main() {

// Устанавливаем кодировку для корректного отображения русского текста

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

// Создаем новый буфер экрана

HANDLE hScreenBuffer = CreateConsoleScreenBuffer(

GENERIC\_READ | GENERIC\_WRITE,

0,

NULL,

CONSOLE\_TEXTMODE\_BUFFER,

NULL

);

if (hScreenBuffer == INVALID\_HANDLE\_VALUE) {

std::wcout << L"Ошибка создания буфера экрана.\n";

return 1;

}

else {

std::wcout << L"Буфер экрана создан успешно.\n";

}

// Активация нового буфера экрана

if (!SetConsoleActiveScreenBuffer(hScreenBuffer)) {

std::wcout << L"Ошибка активации буфера экрана.\n";

CloseHandle(hScreenBuffer);

return 1;

}

else {

std::wcout << L"Буфер экрана активирован.\n";

}

DWORD charsWritten;

COORD pos1 = { 10, 10 }; // Позиция (10, 10)

SetConsoleCursorPosition(hScreenBuffer, pos1);

WriteConsoleW(hScreenBuffer, L"Текст на позиции 10, 10", wcslen(L"Текст на позиции 10, 10"), &charsWritten, NULL);

// Увеличенная задержка для проверки вывода

Sleep(5000);

// Возвращаемся к стандартному буферу экрана перед завершением программы

SetConsoleActiveScreenBuffer(GetStdHandle(STD\_OUTPUT\_HANDLE));

CloseHandle(hScreenBuffer);

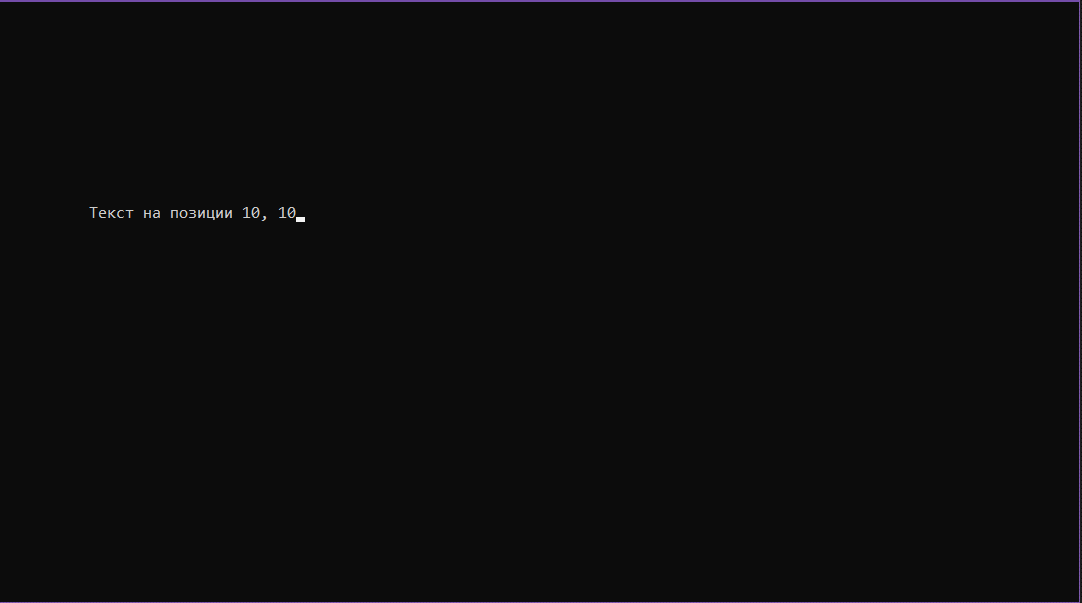
std::wcout << L"Нажмите Enter для завершения...";

std::wcin.get();

return 0;

}

Результат:



Контрольные вопросы

1. Какие функции используются для создания и активации нового буфера экрана?

Для создания нового буфера экрана используется функция CreateConsoleScreenBuffer. Для активации (переключения вывода на созданный буфер) используется функция SetConsoleActiveScreenBuffer.

1. Как можно изменить размер буфера экрана?

Размер буфера экрана можно изменить с помощью функции SetConsoleScreenBufferSize, которая принимает дескриптор буфера и новые размеры (ширину и высоту) в виде структуры COORD.

1. Как изменить положение курсора в буфере экрана?

Положение курсора в буфере экрана изменяется с помощью функции SetConsoleCursorPosition, которая принимает дескриптор буфера и координаты (структура COORD), куда нужно переместить курсор.

1. Каким образом можно установить цвет текста и фона в буфере экрана?

Цвет текста и фона устанавливается с помощью функции SetConsoleTextAttribute, которая принимает дескриптор буфера и атрибуты цвета, комбинируемые с помощью битовых операций. Например, FOREGROUND\_GREEN | BACKGROUND\_BLUE устанавливает зеленый текст на синем фоне.

1. Почему важно возвращаться к стандартному буферу экрана перед завершением программы?

Возвращение к стандартному буферу экрана перед завершением программы важно для обеспечения корректного отображения консоли. Если программа завершится, оставаясь на пользовательском буфере, это может привести к тому, что пользователь не сможет видеть текст или сообщения, так как они будут находиться в неактивном буфере.