### Функции для работы с консолью:

**SetConsoleCursorPosition**

Перемещает курсор в консольном окне в указанную позицию.

void SetCursorPosition(short x, short y) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD position = { x, y };

SetConsoleCursorPosition(hConsole, position);

}

### **Цвета в консоли**:

**SetConsoleTextAttribute**

Устанавливает цвет текста и фона в консоли.

#include <windows.h>

#include <iostream>

using namespace std;

void SetColor(int text, int background) {

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), (background << 4) | text);

}

int main() {

SetColor(4, 0); // Красный текст на черном фоне

cout << "Красный текст" << endl;

SetColor(7, 0); // Вернуть стандартный цвет

cout << "Стандартный текст" << endl;

return 0;

}

* + Черный(0)
  + Синий(1)
  + Зеленый(2)
  + Голубой(3)
  + Красный(4)
  + Фиолетовый(5)
  + Желтый(6)
  + Белый(7)
  + Серый(8)
  + Синий(светлый) (9)
  + Зеленый(светлый) (10)
  + Голубой(светлый) (11)
  + Красный(светлый) (12)
  + Фиолетовый(светлый) (13)
  + Желтый(светлый) (14)
  + Белый(светлый) (15)

### **Перемещение и очистка экрана**:

Для создания анимации в консоли нужно управлять экраном — перемещать курсор и очищать старую графику.

* **Очистка экрана:**

void ClearScreen() {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

DWORD count, cellCount;

COORD homeCoords = { 0, 0 };

if (GetConsoleScreenBufferInfo(hConsole, &csbi)) {

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

FillConsoleOutputCharacter(hConsole, ' ', cellCount, homeCoords, &count);

FillConsoleOutputAttribute(hConsole, csbi.wAttributes, cellCount, homeCoords, &count);

SetConsoleCursorPosition(hConsole, homeCoords);

}

}

* **Перемещение курсора:** Вы можете менять позицию текста для создания эффекта анимации:

void MoveCursor(short x, short y) {

COORD pos = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

}

**4. Анимация в консоли:**

Для создания простой анимации:

1. Перемещайте курсор.
2. Изменяйте текст/символы на новых позициях.
3. Используйте паузы для эффекта движения.

**Пример:**

#include <windows.h>

#include <iostream>

using namespace std;

void MoveCursor(short x, short y) {

COORD pos = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

}

int main() {

for (int i = 0; i < 20; i++) {

MoveCursor(i, 10); // Перемещаемся по строке 10

cout << "\*"; // Печатаем символ

Sleep(100); // Задержка в 100 миллисекунд

MoveCursor(i, 10);

cout << " "; // Стираем символ

}

return 0;

}

### Асинхронный ввод: работа с клавишами

Для обработки нажатий клавиш в режиме реального времени используется функция GetAsyncKeyState. Она позволяет узнать, нажата ли клавиша в данный момент.

Прототип функции: SHORT GetAsyncKeyState(int vKey);

vKey — код клавиши (например, VK\_LEFT для стрелки влево или 'A' для буквы "A").

Возвращает: SHORT, где старший бит показывает, нажата ли клавиша.

Пример использования GetAsyncKeyState:

#include <iostream>

#include <windows.h>

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

cout << "Нажмите клавишу 'A' для продолжения...\n";

while (true) {

if (GetAsyncKeyState('A') & 0x8000) { // Проверка нажатия клавиши 'A'

cout << "Вы нажали 'A'!\n";

break;

}

Sleep(50); // Немного задержки, чтобы снизить нагрузку на процессор

}

return 0;

}

Пример управления объектом с помощью клавиш:

Движение символа в консоли (стрелки клавиатуры):

#include <iostream>

#include <windows.h>

using namespace std;

// Функция перемещения курсора

void MoveCursor(short x, short y) {

COORD pos = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

}

int main() {

short x = 10, y = 10; // Начальные координаты символа

char symbol = '#'; // Символ, который будет двигаться

while (true) {

// Стираем символ с текущей позиции

MoveCursor(x, y);

cout << " ";

// Обрабатываем ввод клавиш

if (GetAsyncKeyState(VK\_UP) & 0x8000) y--; // Стрелка вверх

if (GetAsyncKeyState(VK\_DOWN) & 0x8000) y++; // Стрелка вниз

if (GetAsyncKeyState(VK\_LEFT) & 0x8000) x--; // Стрелка влево

if (GetAsyncKeyState(VK\_RIGHT) & 0x8000) x++; // Стрелка вправо

// Рисуем символ на новой позиции

MoveCursor(x, y);

cout << symbol;

// Выход из цикла по нажатию клавиши ESC

if (GetAsyncKeyState(VK\_ESCAPE) & 0x8000) break;

Sleep(50); // Задержка для плавного движения

}

return 0;

}

### **Полезные константы для клавиш (VK\_):**

Windows API предоставляет константы для большинства клавиш:

* **VK\_UP**: стрелка вверх
* **VK\_DOWN**: стрелка вниз
* **VK\_LEFT**: стрелка влево
* **VK\_RIGHT**: стрелка вправо
* **VK\_ESCAPE**: клавиша Escape
* **VK\_SPACE**: пробел