Что можно сделать в конси

**1. Смена цвета текста и фона**

Для ANSI-кодов:

#include <iostream>

void setTextColor(int color) {

std::cout << "\033[" << color << "m";

}

int main() {

setTextColor(31); // Красный

std::cout << "Это красный текст!" << std::endl;

setTextColor(32); // Зелёный

std::cout << "Это зелёный текст!" << std::endl;

setTextColor(0); // Сброс к стандартному

return 0;

}

**Цвета ANSI-кодов:**

* 30-37: цвет текста (чёрный, красный, зелёный и т.д.).
* 40-47: цвет фона.

Для Windows API:

#include <iostream>

#include <windows.h>

void setConsoleColor(int textColor, int bgColor) {

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

SetConsoleTextAttribute(hConsole, (bgColor << 4) | textColor);

}

int main() {

setConsoleColor(4, 0); // Красный текст, чёрный фон

std::cout << "Красный текст на чёрном фоне" << std::endl;

setConsoleColor(2, 0); // Зелёный текст, чёрный фон

std::cout << "Зелёный текст на чёрном фоне" << std::endl;

setConsoleColor(7, 0); // Сброс к стандартному

return 0;

}

**2. Симуляция кнопок через меню**

Можно создать меню с выбором действий, где каждая "кнопка" — это пункт меню, а переход между "страницами" выполняется через условия.

#include <iostream>

#include <cstdlib> // для очистки экрана

void displayPage1() {

std::cout << "Вы на странице 1!" << std::endl;

std::cout << "1. Перейти на страницу 2" << std::endl;

std::cout << "2. Выход" << std::endl;

}

void displayPage2() {

std::cout << "Вы на странице 2!" << std::endl;

std::cout << "1. Вернуться на страницу 1" << std::endl;

std::cout << "2. Выход" << std::endl;

}

int main() {

int choice = 0;

while (true) {

std::system("cls"); // Очистка экрана

if (choice == 0) {

displayPage1();

} else if (choice == 1) {

displayPage2();

}

std::cin >> choice;

if (choice == 2) break; // Выход

else if (choice == 1 && choice == 0) continue;

}

return 0;

}

**3. Реализация "страниц"**

Страницы можно переключать, изменяя отображаемую часть программы в зависимости от выбора пользователя.

#include <iostream>

void showMainMenu() {

std::cout << "Главное меню:" << std::endl;

std::cout << "1. Открыть страницу 1" << std::endl;

std::cout << "2. Открыть страницу 2" << std::endl;

std::cout << "3. Выход" << std::endl;

}

void showPage1() {

std::cout << "Это страница 1!" << std::endl;

std::cout << "1. Вернуться в главное меню" << std::endl;

}

void showPage2() {

std::cout << "Это страница 2!" << std::endl;

std::cout << "1. Вернуться в главное меню" << std::endl;

}

int main() {

int choice = 0;

while (true) {

std::system("cls"); // Очистка экрана

if (choice == 0) {

showMainMenu();

} else if (choice == 1) {

showPage1();

} else if (choice == 2) {

showPage2();

}

std::cin >> choice;

if (choice == 3) break; // Выход

if (choice != 1 && choice != 2) {

choice = 0; // Возврат в меню

}

}

return 0;

}

**4. Прогресс-бар (дизайн)**

Прогресс-бар можно использовать для визуализации выполнения задачи. Это полезно, например, для загрузки.

#include <iostream>

#include <thread> // Для паузы

#include <chrono> // Для задержки

void showProgressBar(int length) {

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

std::cout << "\r[";

for (int j = 0; j < length; ++j) {

if (j < i) std::cout << "=";

else std::cout << " ";

}

std::cout << "] " << (i \* 100 / length) << "%";

std::cout.flush();

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

}

std::cout << std::endl;

}

int main() {

std::cout << "Загрузка: " << std::endl;

showProgressBar(30); // Прогресс-бар длиной 30 символов

std::cout << "Готово!" << std::endl;

return 0;

}

**5. Анимация текста (дизайн)**

Можно создать эффекты, такие как появление текста или мигание.

#include <iostream>

#include <thread> // Для задержки

#include <chrono>

void typeText(const std::string& text) {

for (char c : text) {

std::cout << c;

std::this\_thread::sleep\_for(std::chrono::milliseconds(50)); // Задержка между символами

}

std::cout << std::endl;

}

int main() {

typeText("Привет! Это текст с эффектом набора.");

return 0;

}

**6. Генерация таблиц**

С помощью форматирования можно выводить таблицы:

#include <iostream>

#include <iomanip> // Для форматирования

void showTable() {

std::cout << std::setw(10) << "Имя"

<< std::setw(10) << "Возраст"

<< std::setw(10) << "Баллы" << std::endl;

std::cout << std::setw(10) << "Иван"

<< std::setw(10) << 20

<< std::setw(10) << 90 << std::endl;

std::cout << std::setw(10) << "Анна"

<< std::setw(10) << 22

<< std::setw(10) << 85 << std::endl;

}

int main() {

showTable();

return 0;

}

**7. Сохранение данных в файл**

Консольные приложения могут сохранять данные для дальнейшего использования.

#include <iostream>

#include <fstream>

int main() {

std::ofstream file("data.txt");

if (file.is\_open()) {

file << "Пример сохранённых данных\n";

file << "Это строка текста." << std::endl;

file.close();

std::cout << "Данные сохранены в файл data.txt." << std::endl;

} else {

std::cout << "Ошибка при открытии файла." << std::endl;

}

return 0;

}

**8. Симуляция текстового редактора (для отзывов допустим)**

Можно создать базовый текстовый редактор, который позволяет вводить текст, редактировать его и сохранять в файл.

#include <iostream>

#include <fstream>

#include <string>

void showMenu() {

std::cout << "Текстовый редактор:" << std::endl;

std::cout << "1. Ввод текста" << std::endl;

std::cout << "2. Показать текст" << std::endl;

std::cout << "3. Сохранить текст в файл" << std::endl;

std::cout << "4. Выход" << std::endl;

}

int main() {

std::string text;

int choice;

while (true) {

showMenu();

std::cout << "Выберите опцию: ";

std::cin >> choice;

std::cin.ignore(); // Убираем лишний Enter из буфера

if (choice == 1) {

std::cout << "Введите текст (пустая строка для завершения ввода):" << std::endl;

std::string line;

while (true) {

std::getline(std::cin, line);

if (line.empty()) break;

text += line + "\n";

}

} else if (choice == 2) {

std::cout << "Ваш текст:" << std::endl;

std::cout << text << std::endl;

} else if (choice == 3) {

std::ofstream file("text.txt");

if (file.is\_open()) {

file << text;

file.close();

std::cout << "Текст сохранён в файл text.txt." << std::endl;

} else {

std::cout << "Ошибка сохранения файла!" << std::endl;

}

} else if (choice == 4) {

break;

} else {

std::cout << "Неверный выбор!" << std::endl;

}

}

return 0;

}

**9. Графический прогресс-бар с вращающейся "загрузкой"**

Добавляем вращающийся символ для эффекта "работы".

#include <iostream>

#include <thread>

#include <chrono>

void showLoadingBar(int steps) {

char spinner[] = {'|', '/', '-', '\\'};

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

std::cout << "\r[";

for (int j = 0; j < steps; ++j) {

if (j < i) std::cout << "#";

else std::cout << " ";

}

std::cout << "] " << spinner[i % 4] << " " << (i \* 100 / steps) << "%";

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

std::cout.flush();

}

std::cout << std::endl;

}

int main() {

std::cout << "Загрузка данных..." << std::endl;

showLoadingBar(30);

std::cout << "Загрузка завершена!" << std::endl;

return 0;

}

**10. Простая база данных**

Создаём консольное приложение, которое позволяет добавлять и выводить данные.

#include <iostream>

#include <vector>

#include <string>

struct Record {

std::string name;

int age;

};

void showMenu() {

std::cout << "1. Добавить запись" << std::endl;

std::cout << "2. Показать все записи" << std::endl;

std::cout << "3. Выход" << std::endl;

}

int main() {

std::vector<Record> records;

int choice;

while (true) {

showMenu();

std::cout << "Ваш выбор: ";

std::cin >> choice;

if (choice == 1) {

Record record;

std::cout << "Введите имя: ";

std::cin >> record.name;

std::cout << "Введите возраст: ";

std::cin >> record.age;

records.push\_back(record);

} else if (choice == 2) {

for (const auto& record : records) {

std::cout << "Имя: " << record.name << ", Возраст: " << record.age << std::endl;

}

} else if (choice == 3) {

break;

} else {

std::cout << "Неверный выбор!" << std::endl;

}

}

return 0;

}

**11. Использование библиотек для запуска процессов**

На современных платформах лучше использовать специализированные библиотеки, например:

* **Boost.Process**: обеспечивает гибкий и кросс-платформенный способ запуска процессов.
* **std::filesystem** (C++17): помогает в работе с путями и файлами.

Пример с использованием Boost.Process:

#include <boost/process.hpp>

#include <iostream>

int main() {

try {

boost::process::child c("other\_program.exe");

c.wait(); // Ожидаем завершения программы

std::cout << "Программа завершена с кодом: " << c.exit\_code() << std::endl;

} catch (const std::exception& e) {

std::cerr << "Ошибка: " << e.what() << std::endl;

}

return 0;

}

Да, в C++ можно запустить один файл с программой из другого. Это достигается с использованием системных вызовов, таких как system, или других методов, доступных через библиотеки. Рассмотрим несколько способов:

**12. Создание DLL для вызова функций**

Если программы связаны друг с другом (например, используют общую логику), можно вынести функциональность в библиотеку (DLL на Windows) и подключать её напрямую через LoadLibrary (Windows).

Пример (Windows):

cpp

Копировать код

#include <windows.h>

#include <iostream>

typedef void (\*FunctionType)(); // Определяем тип функции

int main() {

HINSTANCE hinstLib = LoadLibrary(TEXT("other\_program.dll"));

if (hinstLib) {

FunctionType MyFunction = (FunctionType)GetProcAddress(hinstLib, "FunctionName");

if (MyFunction) {

MyFunction(); // Вызываем функцию из другой программы

} else {

std::cerr << "Не удалось найти функцию в библиотеке." << std::endl;

}

FreeLibrary(hinstLib);

} else {

std::cerr << "Не удалось загрузить DLL." << std::endl;

}

return 0;

}