



9-11 классы

Программирование на C++

Презентация занятия

Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

37 занятие



Минцифры
России



20.35
УНИВЕРСИТЕТ

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Команды

- **Build Solution** () – собрать проект. При этом перекомпилируются все файлы проекта.
- **Rebuild Solution** () – пересобрать проект.
- **Clean Solution** – очистить проект. При этом удаляются все лишние файлы, необходимые на момент разработки и отладки, но не нужные в конечном продукте.
- **Compile** () – скомпилировать проект. При этом перекомпилируются только измененные файлы проекта.
- **Start Debugging** () – начать отладку. Запускает программу под отладчиком.
- **Start without Debugging** () – запустить без отладчика. Просто осуществляется запуск откомпилированной программы.
- **Step Into** () – Пошаговое выполнение с заходом в процедуру.
- **Step Over** () – Пошаговое выполнение без захода в процедуру.
- **Toggle Breakpoint** () – Установить/снять точку останова.
- **Breakpoints** () – показать текущие точки останова.

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Создать простейшее консольное приложение, выводящее на экран фразу приветствия, и проверить его работоспособность.

Создайте приложение *Hello.cpp* (CLR → консольное приложение) и отредактируйте шаблон кода.

```
setlocale(LC_CTYPE, "Russian");
```

Откомпилируйте введенный код. Для этого нажмите на сочетания клавиш *Ctrl+F7* или же в панели меню выберите «*Build=>Compile*» (*Построение => Компилировать*).

После компиляции убедитесь в отсутствии ошибок и запустите полученный exe-файл (в нашем случае он будет называться *Hello.exe* и будет находиться по адресу расположения проекта в папке *Debug*) с помощью сочетания клавиш *Ctrl+F5*.

Заголовочный файл **stdafx.h** должен включаться в файл самым первым.



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Создать простейшее консольное приложение, позволяющее вводить произвольное целое число, увеличивать его на единицу и выводить результат на экран консоли.

Откомпилируйте и запустите введенный код. Проверьте работоспособность программы.

Осуществите трассировку программы , просмотрите значения переменных .

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

В системах, где разрешается перенаправление ввода/вывода, `stdin` и `stdout` можно перенаправлять. Это означает, что они могут быть связаны с устройством, отличным от экрана или клавиатуры. Например, рассмотрим программу:

```
#include <stdio.h>
int main(void)
{
    char str[80];
    printf("Enter a string: ");
    scanf_s("%s", str, 10);
    printf(str);
    return 0;
}
```



ДОП команды для print

Код	Формат
%c	Символ типа char
%d	Десятичное число целого типа со знаком
%i	Десятичное число целого типа со знаком
%e	Научная нотация (е нижнего регистра)
%E	Научная нотация (Е верхнего регистра)
%f	Десятичное число с плавающей точкой
%g	Использует код %e или %f — тот из них, который короче (при использовании %g используется е нижнего регистра)
%G	Использует код %E или %f — тот из них, который короче (при использовании %G используется Е верхнего регистра)
%o	Восьмеричное целое число без знака
%s	Строка символов
%u	Десятичное число целого типа без знака
%x	Шестнадцатичное целое число без знака (буквы нижнего регистра)
%X	Шестнадцатичное целое число без знака (буквы верхнего регистра)
%p	Выводит на экран значение указателя
%n	Ассоциированный аргумент — это указатель на переменную целого типа, в которую помещено количество символов, записанных на данный момент
%%	Выводит символ %





ДОП команды для scan

Код	Значение
%c	Считать один символ
%d	Считать десятичное число целого типа
%i	Считать десятичное число целого типа
%e	Считать число с плавающей запятой
%f	Считать число с плавающей запятой
%g	Считать число с плавающей запятой
%o	Считать восьмеричное число
%s	Считать строку
%x	Считать шестнадцатиричное число
%p	Считать указатель
%n	Принимает целое значение, равное количеству считанных до текущего момента символов
%u	Считывает беззнаковое целое
%[]	Просматривает набор символов
%%	Считывает символ %



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Предположим, что данная программа называется TEST. При обычном выполнении TEST она выводит на экран подсказку, читает с клавиатуры строку и выводит данную строку на экран. Как stdin, так и stdout могут быть перенаправлены в файл. Например
TEST > OUTPUT

вывод программы TEST будет происходить в файл OUTPUT. Выполнение команды
TEST < INPUT > OUTPUT

приведет к тому, что программа будет читать из файла INPUT, а записывать в файл OUTPUT.

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Функция `SetConsoleTextAttribute` устанавливает атрибуты цвета текста (цвет букв и цвет фона ячеек с буквами). Она использует следующий синтаксис `BOOL SetConsoleTextAttribute(HANDLE,WORD);` и принимает два аргумента, обозначающие дескриптор окна (в нашем случае консоли), и число, биты которого определяют цвета



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Для получения идентификатора/дескриптора консольного окна (HANDLE) можно использовать функцию `GetStdHandle`: `HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);`
Здесь константа `STD_OUTPUT_HANDLE` говорит, что вы хотите получить дескриптор стандартного устройства ввода/вывода `stdout`. Вторым аргументом в функцию `SetConsoleTextAttribute` передается число, биты которого управляют цветами фона и текста. Например, команды `SetConsoleTextAttribute(hStdOut, 2); cout << "Green\n";` выводят текст зеленым цветом на черном фоне, где `hStdOut` должен быть получен, так как описано ранее.

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Для получения идентификатора/дескриптора консольного окна (HANDLE) можно использовать функцию `GetStdHandle`: `HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);`
Здесь константа `STD_OUTPUT_HANDLE` говорит, что вы хотите получить дескриптор стандартного устройства ввода/вывода `stdout`. Вторым аргументом в функцию `SetConsoleTextAttribute` передается число, биты которого управляют цветами фона и текста. Например, команды `SetConsoleTextAttribute(hStdOut, 2); cout << "Green\n";` выводят текст зеленым цветом на черном фоне, где `hStdOut` должен быть получен, так как описано ранее.

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Второй аргумент функции `SetConsoleTextAttribute` использует младший байт слова `WORD`. Т.о. цвета фона и текста вы можете задать, передав вторым аргументом число в диапазоне от 0 до 255. Его биты будут управлять наличием или отсутствием RGB составляющих в цвете фона и текста.

Однако второй аргумент функции `SetConsoleTextAttribute` удобнее конструировать из набора флагов. Интересующие нас флаги включают `BACKGROUND_` и `FOREGROUND_`, которые обозначают цвет фона и текста соответственно. Например, команды `SetConsoleTextAttribute(hStdOut, FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_INTENSITY); cout << "Yellow\n";`





Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

Аналогично задаются цвета фона. Например, команды `SetConsoleTextAttribute(hStdOut, BACKGROUND_BLUE | BACKGROUND_INTENSITY | FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE | FOREGROUND_INTENSITY);` `cout << "White on blue\n";` выводят белый текст на ярко синем фоне.



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <windows.h>
#include <iostream>
using namespace std;
int main()
{
    //получаем дескриптор
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    //Выводим текст разными цветами
    SetConsoleTextAttribute(hStdOut, 2);
    cout << "Green\n"; //зеленый текст на черном фоне
    SetConsoleTextAttribute(hStdOut, FOREGROUND_RED |
        FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    cout << "Yellow\n"; //желтый текст на черном фоне
    SetConsoleTextAttribute(hStdOut,
        BACKGROUND_BLUE | BACKGROUND_INTENSITY |
        FOREGROUND_RED | FOREGROUND_GREEN |
        FOREGROUND_BLUE | FOREGROUND_INTENSITY);
    cout << "White on blue\n"; //белый текст на синем фоне
}
```



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
C:\Users\parrot\source\repos\ConsoleApplication7\x64\Debug>
```

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Для продолжения нажмите любую клавишу . . .





Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
#define _CRT_SECURE_NO_WARNINGS  
#include <windows.h>  
#include <string>  
#include <iostream>
```

```
using namespace std;  
string intto4chars(int val1, int val2);
```



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

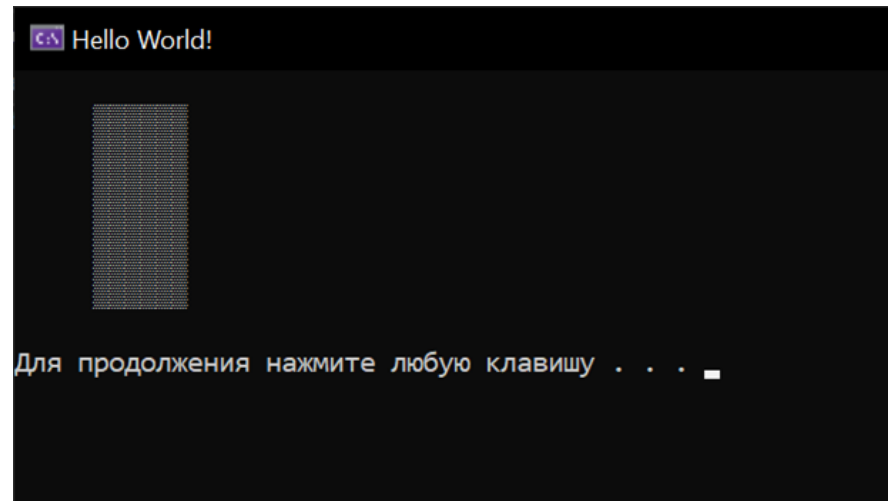
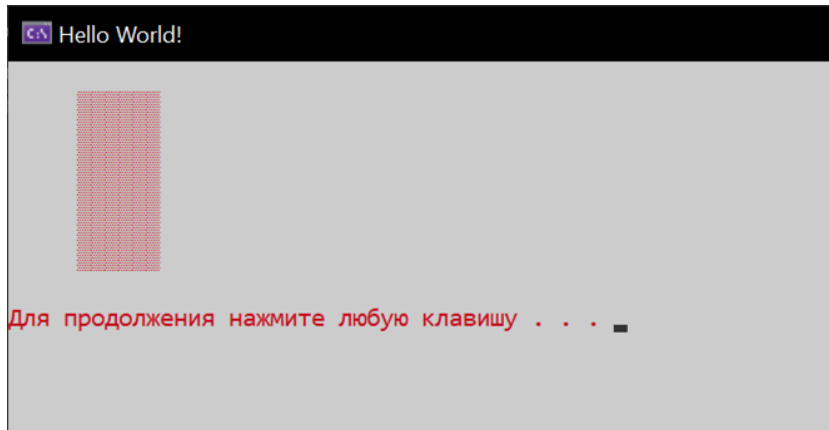
```
int main()
{
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hStdOut, BACKGROUND_RED |
    BACKGROUND_GREEN | BACKGROUND_BLUE);
    system("CLS"); // заливка консольного окна серым цветом
    char strbox[] = " "; //восемь пробелов
    int k, IR, GB, rez;
    string strrez;
    cout << "\n\n";
    for (IR = 0; IR <= 3; IR++)
    { for (k = 0; k < 3; k++)
      { for (GB = 0; GB <= 3; GB++)
        { strrez = intto4chars(IR, GB);
          rez = (IR << 6) + (GB << 4);
          SetConsoleTextAttribute(hStdOut, rez);
          if (k == 1) cout << " " << strrez << "\t";
          else cout << strbox;}
        cout << "\n";}}
    cout << "\n";
    SetConsoleTextAttribute(hStdOut, 0x70);
    system("pause");}
```

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
string intto4chars(int val1, int val2)
{
    char chrez[20];
    int len = 4;
    int val = (val1 << 2) + val2;
    string sbfull(len, '0');
    string sbrez(_itoa(val, chrez, 2));
    sbfull.replace(len - sbrez.length(), sbrez.length(), sbrez);
    return sbfull;
}
```



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++



Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
CONSOLE_SCREEN_BUFFER_INFO csbInfo;  
void drawBox(HANDLE hStdOut, COORD Pos);  
int main()  
{ SetConsoleTitle(L"Hello World!"); //заголовок окна  
HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);  
GetConsoleScreenBufferInfo(hStdOut, &csbInfo);  
WORD wOldColorAttrs;  
wOldColorAttrs = csbInfo.wAttributes; // старые атрибуты  
SetConsoleTextAttribute(hStdOut, BACKGROUND_RED |  
BACKGROUND_GREEN | BACKGROUND_BLUE | FOREGROUND_RED);  
system("CLS"); // заливка консольного окна серым цветом  
COORD cursorPos = { 5,1 }; // к-ты левого верхнего угла  
drawBox(hStdOut, cursorPos); // Печать прямоугольника  
cursorPos = { 0,8 };  
SetConsoleCursorPosition(hStdOut, cursorPos);  
system("pause"); // Возвращаем исходные цвета  
SetConsoleTextAttribute(hStdOut, wOldColorAttrs);  
system("CLS"); //очищаем/перекрашиваем все окно  
cursorPos = { 5,1 };  
drawBox(hStdOut, cursorPos); // Снова рисуем прямоугольник  
cursorPos = { 0,8 }; // Новое положение текстового курсора  
SetConsoleCursorPosition(hStdOut, cursorPos);  
system("pause");}
```





Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
void drawBox(HANDLE hStdOut, COORD Pos)
{
    for (int i = 0; i < 6; i++) {
        SetConsoleCursorPosition(hStdOut, Pos);
        for (int j = 0; j < 6; j++) cout << char(177);
        Pos.Y += 1;
    }
}
```



Задание 1

A dark gray rectangular area, likely a drawing canvas. In the top-left corner, there is a small white square. To its right, the text "C:5 Draw block" is displayed in a light gray font. The rest of the area is empty.

Тема: Консольные приложения: изучение основных приемов работы в среде Microsoft Visual C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <conio.h>
#include <iostream>
#include <windows.h>
using namespace std;
int main() {
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
SetConsoleTitle(L"Draw block");
const int row = 10; // количество строк блока
const int col = 10; // количество столбцов блока
CHAR_INFO consBuffer[col * row]; // массив блока данных
WORD attrib = FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_INTENSITY; //желтый текст на черном фоне // заполнение массива блока данных
for (int i = 0; i < col * row; ++i) {
consBuffer[i].Char.UnicodeChar = 'X';
consBuffer[i].Attributes = attrib;}
COORD charPosition = { 0,0 }; // точка в блоке
SMALL_RECT writeArea = { 10,5,19,14 };
COORD bufferSize = { col,row };
WriteConsoleOutput(hConsole, consBuffer, bufferSize,
charPosition, &writeArea);
_getch();
}
```