

SARFTI

GAMEDEV LAB

C++

Часть 1.

Что понадобится?

ОС Windows 10

MS Visual Studio 2019

Зарегистрированный аккаунт
Microsoft

Желание экспериментировать)

SARFTI

GAMEDEV LAB



Visual
Studio



Создание первой(консольной) программы – запуск VS

Недавние проекты

Visual Studio 2019

Открыть последние

Сегодня

- | | |
|---|------------------|
|  TestConsole.sln | 13.04.2020 18:24 |
| C:\Users\povel\source\repos\TestConsole | |
|  Test.sln | 13.04.2020 18:21 |
| C:\Users\povel\source\repos\Test | |

Начало работы

-  Копирование или извлечение кода
Получить код из интернет-репозитория, например, GitHub или Azure DevOps
-  Открыть проект или решение
Открыть локальный проект Visual Studio или SLN-файл
-  Открыть локальную папку
Перейти и изменить код в любой папке
-  Создание проекта
Выберите шаблон проекта с формированием шаблонов кода, чтобы начать работу

Нам сюда

[Продолжить без кода →](#)

Создание первой(консольной) программы – шаблоны проектов

Создание проекта

Последние шаблоны проектов

-  Консольное приложение C++
-  Пустой проект C++

Поиск шаблонов (ALT+“В”)

Очистить все

C++

Windows

Консоль



Пустой проект

Начать с нуля, используя C++ для Windows. Начальные файлы отсутствуют.

C++ Windows Консоль



Консольное приложение

Выполнить код в терминале Windows. По умолчанию выводится фраза "Hello World".

C++ Windows Консоль



Мастер классических приложений Windows

Создание собственного приложения Windows с помощью мастера.

C++ Windows Рабочий стол Консоль Библиотека



Проект общих элементов

Проект общих элементов используется для совместного использования файлов в нескольких проектах.

C++ Windows Android iOS Linux Рабочий стол

Консоль Библиотека UWP Игры Мобильный



Проект CMake

Создавайте современные кроссплатформенные приложения C++, не зависящие от файлов SLN или VCXPROJ.

Назад

Далее

Выбираем

Создание первой(консольной) программы – название проекта и расположение

Настроить новый проект

Консольное приложение C++ Windows Консоль

Имя проекта

ConsoleApplication1

Расположение

C:\Users\pove\source\repos

Имя решения 

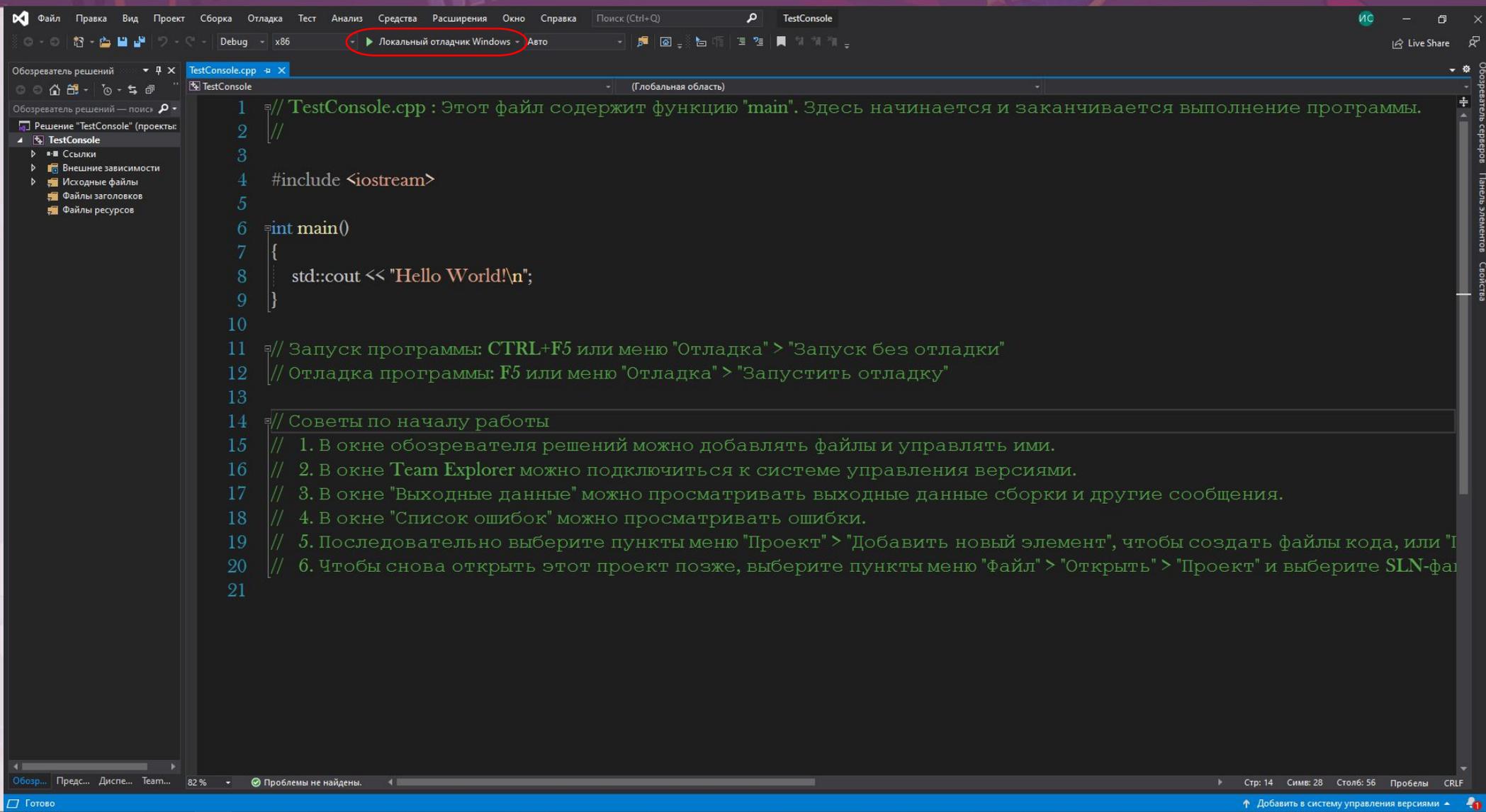
ConsoleApplication1

Поместить решение и проект в одном каталоге

Назад

Создать

Создание первой(консольной) программы – запуск отладчика



```
1 // TestConsole.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
2 //
3
4 #include <iostream>
5
6 int main()
7 {
8     std::cout << "Hello World!\n";
9 }
10
11 // Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"
12 // Отладка программы: F5 или меню "Отладка" > "Запустить отладку"
13
14 // Советы по началу работы
15 // 1. В окне обозревателя решений можно добавлять файлы и управлять ими.
16 // 2. В окне Team Explorer можно подключиться к системе управления версиями.
17 // 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.
18 // 4. В окне "Список ошибок" можно просматривать ошибки.
19 // 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить новый проект", чтобы создать новый проект.
20 // 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл проекта.
21
```

F5

Настройка VS

The image shows a screenshot of the Visual Studio IDE interface. The top menu bar includes 'Файл', 'Правка', 'Вид', 'Проект', 'Сборка', 'Отладка', 'Тест', and 'Анализ'. The 'Tools' menu is open, displaying various options. The 'Parameters...' option at the bottom is circled in red. The background shows a code editor with C++ code for 'TestConsole.cpp' and a Solution Explorer on the left.

Меню 'Средства' (Tools) содержит следующие пункты:

- Получить средства и компоненты...
- Подключиться к базе данных...
- Подключиться к серверу...
- Диспетчер фрагментов кода... (Ctrl+K, Ctrl+B)
- Выбрать элементы панели элементов...
- Диспетчер пакетов NuGet
- Python
- Создать GUID
- Поиск ошибки
- Spy++
- ILDasm
- Командная строка Visual Studio
- Внешние инструменты...
- Командная строка
- Импорт и экспорт параметров...
- Настройка...
- Параметры...** (выделено красным)

Видимый код в редакторе:

```
1 // TestConsole
2 //
3
4 #include <iostream>
5
6 int main()
7 {
8     std::cout << "Hello, World!" << endl;
9 }
10
11 // Запуск программы: CTRL+F5 или меню "Отл"
```

Настройка VS - визуализация

Выбираем удобную
вам тему

Параметры

? X

Параметры поиска (CTRL+E)

- Окружение
 - Общие**
 - Автовосстановление
 - Веб-браузер
 - Вкладки и окна
 - Выбор языка
 - Документы
 - Запуск
 - Импорт и экспорт параметров
 - Клавиатура
 - Найти и заменить
 - Обновления продукта
 - Параметры доверия
 - Расширения
 - Список задач
 - Учетные записи

Визуальное представление

Цветовая тема: Темная

- Применить регистр заголовка к строке меню
- Оптимизировать отображение для экранов с другой плотностью пикселей (требуется перезапуск)
- Автонастройка представления в зависимости от быстродействия клиента
 - Включить расширенное визуальное представление клиента
 - Использовать аппаратное ускорение обработки изображения, если доступно

Сейчас Visual Studio использует аппаратное ускорение отрисовки. Этот режим изменяется автоматически в зависимости от возможностей системы.

Элементы для отображения в меню "Окно": 10

Элементы для отображения в списке недавно использованных файлов: 10

- Отображать строку состояния (требуется перезапуск)
- Кнопка закрытия действует только на активное окно
- Кнопка автоскрытия действует только на активное окно

OK

Отмена

Параметры

▲ Окружение

Общие

Автосохранение

Веб-браузер

Вкладки и окна

Выбор языка

Документы

Сохранять данные автовосстановления каждые:

5 МИН.

Хранить данные автовосстановления:

7 ДН.

Настройка VS - автосохранение

Параметры

Параметры поиска (CTRL+E)

- Окружение
 - Общие
 - Автосохранение
 - Веб-браузер
 - Вкладки и окна
 - Выбор языка
 - Документы
 - Запуск
 - Импорт и экспорт параметров
 - Клавиатура
 - Найти и заменить
 - Обновления продукта
 - Параметры доверия
 - Расширения
 - Список задач
 - Учетные записи
 - Функции предварительной верс
 - Шрифты и цвета**
 - Проекты и решения
 - Рабочие элементы
 - Система управления версиями
 - Текстовый редактор
 - Отладка
 - Средства производительности

Параметры для:

Текстовый редактор По умолчанию

Шрифт (моноширинные шрифты имеют полужирное начертание): Consolas Размер: 24

Отображаемые элементы:

- Обычный текст**
- Выделенный текст
- Неактивный выделенный текст
- Поле индикаторов
- Номер строки
- Видимое пустое пространство
- Code Review Comments
- Code Review Comments (filtered)
- Comment Highlight
- Comment Mark
- DML-маркер SQL
- NAT-бит регистра
- Python Comma
- Python Dot
- TMClassifier
- Автоматически подставлена скобка
- Адрес памяти
- Активный оператор
- Атрибут XAML
- Атрибут XML
- Атрибут вспомогательной функции HTML-тегов Razor
- Атрибут разметки
- Вернуть новый контекст

Основной цвет элемента: По умолчанию Другой...

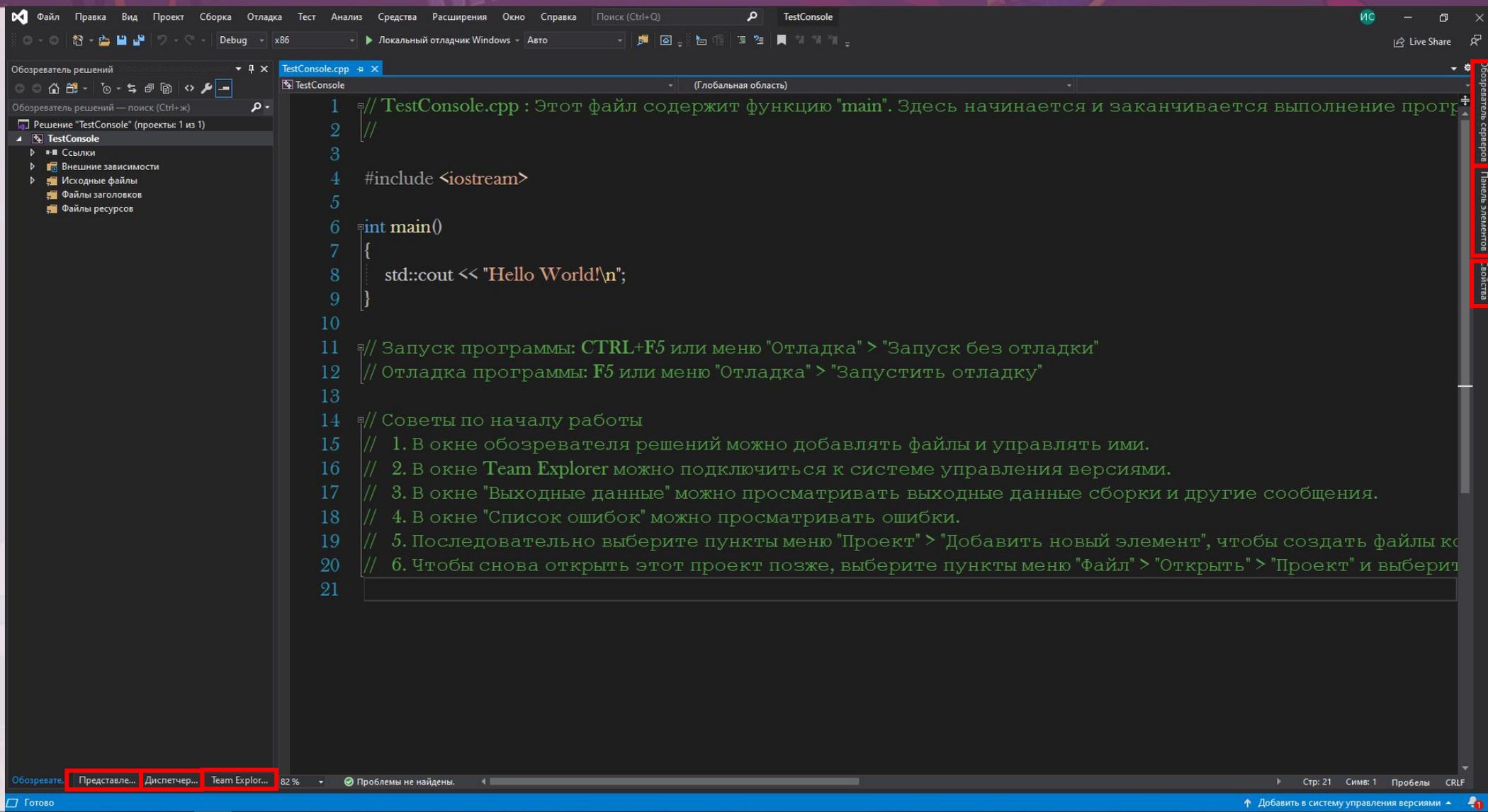
Фоновый цвет элемента: По умолчанию Другой...

Полужирный

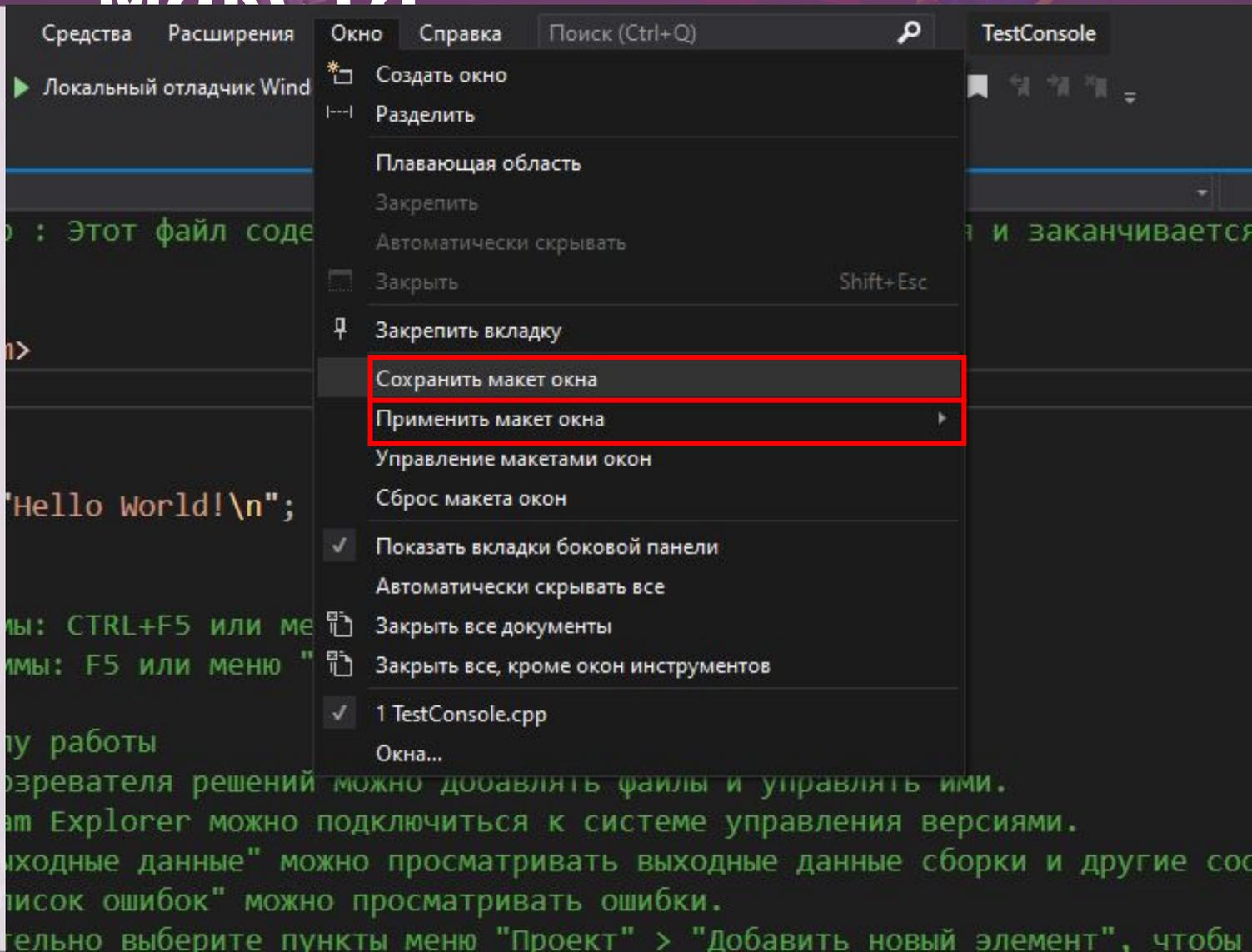
Образец:
 = I :: o0(0xB81

OK Отмена

Настройка макета окна – закрываем лишнее



Настройка макета окна – сохранение и применение макета



Переменные – фундаментальные типы

Логический тип

`bool` — тип, способный хранить одно из двух значений: `true` (истина) или `false` (ложь).

Символьные типы

`signed char` — тип для знакового представления символов.

`unsigned char` — тип для беззнакового представления символов.

`char` — тип для представления символов, который может наиболее эффективно обрабатываться в целевой системе (эквивалентный `signed char` или `unsigned char`, но всё же отличный от них тип).

`wchar_t` — тип для широкого представления символов.

`char16_t` — тип для представления символов в UTF-16. (начиная с C++11)

`char32_t` — тип для представления символов в UTF-32. (начиная с C++11)

Целочисленные типы

`int` — базовый целочисленный тип. Может быть опущен, если представлен любой из модификаторов. Если не представлен ни один из модификаторов размера, гарантировано имеет ширину не меньше 16 бит. Тем не менее, на 32/64-битных системах почти всегда имеет ширину не меньше 32 бит (см. ниже).

Модификаторы

Модифицируют целочисленный тип. Могут располагаться в любом порядке. Только один модификатор из каждой группы может быть представлен в имени типа.

Знаковость

`signed` — целевой тип будет иметь знаковое представление (по умолчанию, если не представлен ни один из вариантов).

`unsigned` — целевой тип будет иметь беззнаковое представление.

Размер

`short` — целевой тип будет оптимизирован по размеру и иметь ширину не меньше 16 бит.

`long` — целевой тип будет иметь ширину не меньше 32 бит.

`long long` — целевой тип будет иметь ширину не меньше 64 бит. (начиная с C++11)

ТВОЕ СВОБОДНОЕ ВРЕМЯ
КАК КОТ ШРЕДИНГЕРА

ОНО ЕСТЬ И В ТОЖЕ
ВРЕМЯ НЕТ

Типы с плавающей точкой

`float` — тип с плавающей точкой одинарной точности. Обычно 32-битный тип с плавающей точкой формата IEEE-754

`double` — тип с плавающей точкой двойной точности. Обычно 64-битный тип с плавающей точкой формата IEEE-754

`long double` — тип с плавающей точкой повышенной точности. Не обязательно отображается на типы IEEE-754. Обычно 80-битный тип с плавающей точкой формата x87 на архитектурах x86 и x86-64.

Переменные – фундаментальные типы

Наиболее часто используемые:

- Логический тип: bool
- Символьный тип: char
- Целочисленный тип: int
- С плавающей запятой: float, double

Тип	Размер в битах	Формат	Промежуток значений	
			Приблизительный	Точный
символьный	8	знаковый (обратный код)	от -127 до 127	
		знаковый (дополнительный код)	от -128 до 127	
		беззнаковый	от 0 до 255	
целочисленный	16	знаковый (обратный код)	$\pm 3,27 \cdot 10^4$	от -32767 до 32767
		знаковый (дополнительный код)		от -32768 до 32767
		беззнаковый	от 0 до $6,55 \cdot 10^4$	от 0 до 65535
	32	знаковый (обратный код)	$\pm 2,14 \cdot 10^9$	от -2147483647 до 2147483647
		знаковый (дополнительный код)		от -2147483648 до 2147483647
		беззнаковый	от 0 до $4,29 \cdot 10^9$	от 0 до 4294967295
	64	знаковый (обратный код)	$\pm 9,22 \cdot 10^{18}$	от -9223372036854775807 до 9223372036854775807
		знаковый (дополнительный код)		от -9223372036854775808 до 9223372036854775807
		беззнаковый	от 0 до $1,84 \cdot 10^{19}$	от 0 до 18446744073709551615
с плавающей точкой	32	IEEE-754 ↗	$\pm 3,4 \cdot 10^{\pm 38}$ (~7 цифр)	<ul style="list-style-type: none"> • мин. денормализованное: $\pm 1,4012984 \cdot 10^{-47}$ • мин. нормализованное: $\pm 1,1754943 \cdot 10^{-38}$ • макс.: $\pm 3,4028234 \cdot 10^{38}$
	64	IEEE-754	$\pm 1,7 \cdot 10^{\pm 308}$ (~15 цифр)	<ul style="list-style-type: none"> • мин. денормализованное: $\pm 4,940656458412 \cdot 10^{-324}$ • мин. нормализованное: $\pm 2,2250738585072014 \cdot 10^{-308}$ • макс.: $\pm 1,7976931348623157 \cdot 10^{308}$

Переменные – дополнительные ТИПЫ

void – указывает на отсутствие информации



enum – используется для перечисления чего-либо

```
int count = 10 ; // Присваиваем переменной count
                  // начальное значение 10

char ch = 'X1' ; // Инициализируем ch буквой X

float f = 1.2F   // Переменная f инициализируется
                  // числом 1.2

int a,b=8,c=19,d; // Переменные b и c
                  // инициализируются числами.
```

Переменные – правила составления имён переменных

- Названия переменных начинаются с нижнего регистра
Пример: `int countMems = 125;`
- Нельзя использовать специальные символы в названии(пробел, запятые и др.).
- Имя переменной не должно совпадать с ключевыми словами(такие как `function`, `char` и др.).
- Все имена следует записывать на английском
Нельзя: `char bukovka = "А";`
- Если имя переменной состоит более чем из одного слова, то имя записывается без пробелов, а все слова после первого начинаются с верхнего регистра.
Пример: `string londonIsTheCapitalOfGreatBritain;`



Особенности типов: bool

Переменные логического типа **bool** могут принимать лишь два значения — **true** (истина) или **false** (ложь). Логические переменные служат для хранения результатов логических операций. Например:

```
void f(int a, int b)
{
    bool bl = a==b; // = есть присваивание, == проверка на равенство;
    // ...
}
```

По определению, при преобразовании к типу **int** значению **true** сопоставляется **1**, а значению **false** — **0**. В обратном направлении целые значения неявно преобразуются в логические следующим образом: ненулевые значения трактуются как **true**, а **0** как **false**. Например:

```
bool b=7; // bool(7) есть true, так что b инициализируется значением true
int i= true; // int(true) есть 1, так что i инициализируется значением 1
```

```
bool x=a+b; // a+b равно 2, так что x становится true
bool y=a | b; // a|b равно 1, так что y становится true
```

```
}
```

Особенности типов: char

В переменной типа *char* может храниться код символа, соответствующий некоторой стандартной кодировке. Например:

```
char ch = 'a' ;
```

Каждому символу сопоставляется целочисленное значение. Например, символу *'b'* в рамках кодировки ASCII соответствует число 98. Вот программа, которая ответит на вопрос о числовом коде любого символа, который вы введете с клавиатуры:

```
#include <iostream>  
  
int main ()  
{  
    char c;  
    std::cin >> c;  
    std::cout << "the value of '" << c << "' is " << int(c) << '\n' ;  
}
```

типы данных с плавающей точкой (ПЗ)

float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

Особенности типов: int

Как и тип *char*, каждый целый тип представим в трех формах: «просто» *int*, *signed int* и *unsigned int*. Кроме того, целые могут быть трех размеров: *short int*, «просто» *int* и *long int*. Вместо *long int* можно просто писать *long*. Аналогично, *short* есть синоним для *short int*, *unsigned* — для *unsigned int*, а *signed* — для *signed int*.

Тип	байт	Диапазон принимаемых значений
целочисленные типы данных (ФЗ)		
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	4	-2 147 483 648 / 2 147 483 647
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295

Спецификатор типа	Эквивалентный тип	Ширина в битах
		Стандарт C++
short	short int	не меньше чем 16
short int		
signed short		
signed short int		
unsigned short	unsigned short int	
unsigned short int		
int	int	не меньше чем 16
signed		
signed int		
unsigned	unsigned int	
unsigned int		
long	long int	не меньше чем 32
long int		
signed long		
signed long int		
unsigned long	unsigned long int	
unsigned long int		
long long	long long int (C++11)	не меньше чем 64
long long int		
signed long long		
signed long long int		
unsigned long long	unsigned long long int (C++11)	
unsigned long long int		

Особенности типов: double

Типы с плавающей запятой соответствуют числам с плавающей запятой. Как и целые типы, они могут быть *трех размеров*: **float** (одинарная точность), **double** (двойная точность) и **long double** (расширенная точность).

Точный смысл каждого типа зависит от реализации. Выбор оптимальной точности в конкретных задачах требует изрядных знаний в области вычислений с плавающей запятой. Если у вас их нет, то проконсультируйтесь со специалистом, или основательно изучите предмет, или используйте тип **double** наудачу.

с плавающей точкой	32	IEEE-754 ↗	$\pm 3,4 \cdot 10^{\pm 38}$ (~7 цифр)	<ul style="list-style-type: none"> • мин. денормализованное: $\pm 1,4012984 \cdot 10^{-47}$ • мин. нормализованное: $\pm 1,1754943 \cdot 10^{-38}$ • макс.: $\pm 3,4028234 \cdot 10^{38}$
	64	IEEE-754	$\pm 1,7 \cdot 10^{\pm 308}$ (~15 цифр)	<ul style="list-style-type: none"> • мин. денормализованное: $\pm 4,940656458412 \cdot 10^{-324}$ • мин. нормализованное: $\pm 2,2250738585072014 \cdot 10^{-308}$ • макс.: $\pm 1,7976931348623157 \cdot 10^{308}$

$1 \equiv \text{sizeof}(\text{char}) \leq \text{sizeof}(\text{short}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long})$
 $1 \leq \text{sizeof}(\text{bool}) \leq \text{sizeof}(\text{long})$
 $\text{sizeof}(\text{char}) \leq \text{sizeof}(\text{wchar}_t) \leq \text{sizeof}(\text{long})$
 $\text{sizeof}(\text{float}) \leq \text{sizeof}(\text{double}) \leq \text{sizeof}(\text{long double})$
 $\text{sizeof}(N) \equiv \text{sizeof}(\text{signed } N) \equiv \text{sizeof}(\text{unsigned } N)$

Объявление и инициализация

Объявление состоит из четырех частей: необязательного «спецификатора», базового типа, *декларатора (declarator)* и необязательного *инициализирующего выражения (инициализатора — initializer)*. За исключением определений функции и пространства имен объявление заканчивается точкой с запятой. Например:

```
char* kings [] = { "Antigonus", "Seleucus", "Ptolemy" };
```

*	указатель	<i>prefix</i>
*const	константный указатель	<i>prefix</i>
&	ссылка	<i>prefix</i>
[]	массив	<i>postfix</i>
()	функция	<i>postfix</i>

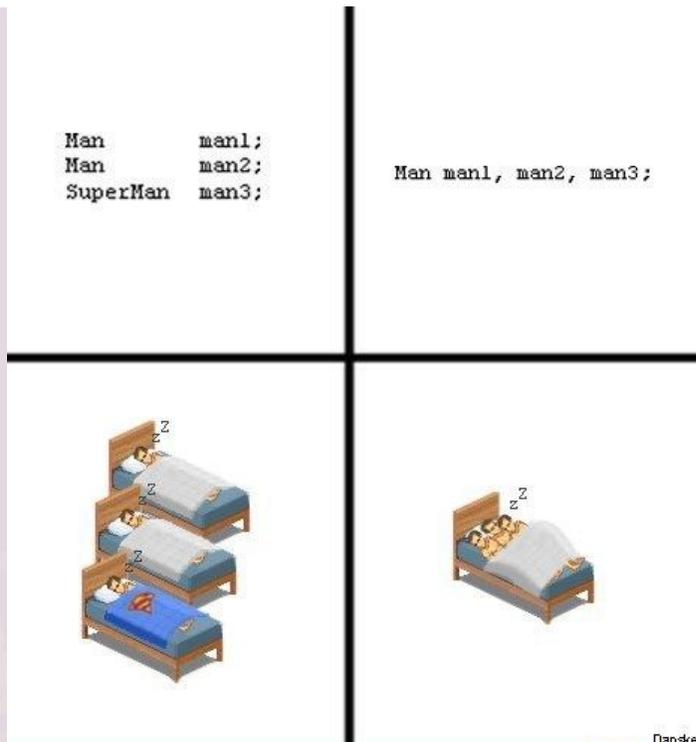
```
int* p, y; // int* p; int y; (не int* y;)
```

```
int x, *q; // int x; int* q;
```

```
int v[10], *pv; // int v[10]; int* pv;
```

```
int a; // означает int a = 0;
```

```
double d; // означает double d = 0.0;
```



Задание №1

Инициализировать переменные каждого из фундаментальных типов и присвоить им соответствующие значения.

Наиболее часто используемые:

- Логический тип: bool
- Символьный тип: char
- Целочисленный тип: int
- С плавающей запятой: float, double

Перечисление (*enumeration*) является *типом*, содержащим набор значений, определяемых пользователем (программистом). После определения перечисление используется почти так же, как и целые типы.

Именованные целые константы служат элементами перечислений. Например, следующее объявление

```
enum {ASM, AUTO, BREAK};
```

Каждое перечисление является отдельным типом. *Типом элемента перечисления* является *тип самого перечисления*. Например, тип *AUTO* есть *keyword*.

```
enum e1 {dark, light}; // диапазон 0:1
enum e2 {a = 3, b = 9}; // диапазон 0:15
enum e3 {min = -10, max = 1000000}; // диапазон -1048576:1048575
enum flag {x=1, y=2, z=4, e=8 }; // диапазон 0:15

flag f1=5; // ошибка типа: 5 не принадлежит к типу flag
flag f2=flag (5); // ок: flag(5) принадлежит типу flag и входит в его диапазон
flag f3=flag (z|e); // ок: flag(12) принадлежит типу flag и входит в его диапазон
flag f4=flag (99); // не определено: 99 не входит в диапазон типа flag
```

```
void f(keyword key)
{
    switch (key)
    {
        case ASM:
            // некоторые действия
            break;
        case BREAK:
            // некоторые действия
            break;
    }
}
```

Задание №2

Создайте перечисление Zoo с 3-4 элементами, где в качестве имён элементов будут выступать какие-нибудь животные с различным количеством конечностей, а значение элемента равно этому количеству конечностей.

```
enum e1 { dark, light } ;  
enum e2 { a = 3, b = 9 } ;  
enum e3 { min = -10, max = 1000000 } ;
```

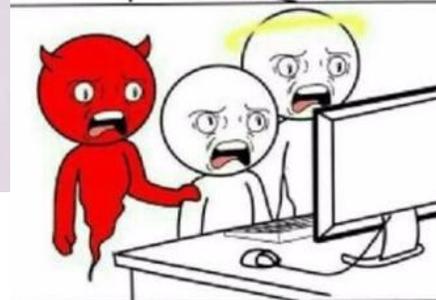
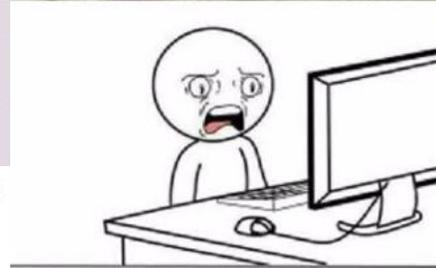
typedef

Объявление, начинающееся с ключевого слова **typedef**, определяет новое имя для типа, а не новую переменную какого-либо существующего типа. Например:

```
typedef char* Pchar;
Pchar p1, p2;      // p1 и p2 типа char*
char* p3=p1;
```

```
typedef int int32;
typedef short int16;
```

Если теперь использовать тип **int32** всюду, где нужны большие целые, то программу будет легко портировать на системную платформу, для которой **sizeof(int)** равен 2. Действительно, нужно лишь дать новое определение для **int32**:



```
5 namespace = std;
6 using = int;
7 using = void;
8 using = time_t;
9 using = bool;
10 #define auto
11 #define enum
12 #define false
13 #define true
14 #define "evil"
15 #define ::make_shared
16 #define virtual
17 #define ::cout
18 #define ::endl
19 template<class >
20 using = ::vector<>;
21 template<class >
22 using = ::shared_ptr<>;
23
24 { , , , };
25 () { return ::rand(); }
26 () { return ; }
27
28 struct || { I ; () = 0; };
29 struct : || { I ; () { << " " << ; }; };
30 struct : || { I ; () { << " " << ; }; };
31 struct : || { I ; () { << " " << ; }; };
32 struct : || { I ; () { << " " << ; }; };
33 struct : || { I ; () { << " " << ; }; };
34 struct : || { I ; () { << " " << ; }; };
35
36 main()
37 {
38     if ( () == )
39         << " " << ;|
40
41     << <|>> = { <>(), <>(), <>(), <>(), <>() };
42
43     for ( : )
44         ->();
45
46     return ();
47 }
48
```

Задача №1

Напишите программу, которая выводит буквы 'a' – 'z' и цифры '0' – '9' и их десятичные коды. Повторите всё для иных символов, имеющих зрительные образы.

Задача №2

Напишите программу, которая выводит размеры типов данных: `char`, `short`, `int`, `long`, `float`, `double`, `long double` и `unsigned`.

Все программы сделайте в одном проекте, который создали в начале занятия.

