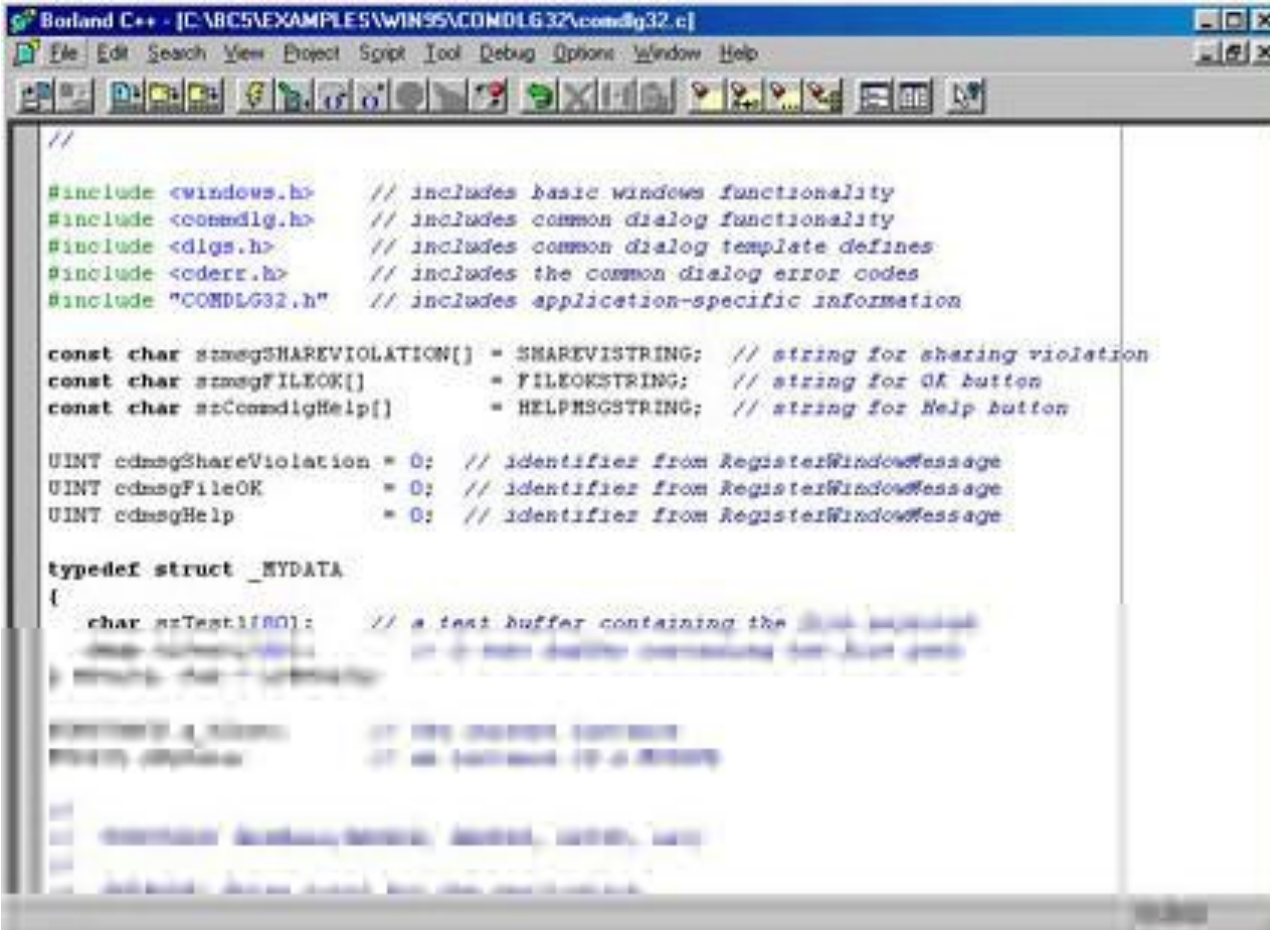


среда программирования Windows.



```

//
#include <windows.h> // includes basic windows functionality
#include <commdlg.h> // includes common dialog functionality
#include <dlls.h> // includes common dialog template defines
#include <cderr.h> // includes the common dialog error codes
#include "COMDLG32.H" // includes application-specific information

const char szMsgSHAREVIOLATION[] = SHAREVISTRING; // string for sharing violation
const char szMsgFILEOK[] = FILEOKSTRING; // string for OK button
const char szCmdDlgHelp[] = HELPHSGSTRING; // string for Help button

UINT cdMsgShareViolation = 0; // identifier from RegisterWindowMessage
UINT cdMsgFileOK = 0; // identifier from RegisterWindowMessage
UINT cdMsgHelp = 0; // identifier from RegisterWindowMessage

typedef struct _MYDATA
{
    char szTest[100]; // a test buffer containing the test message
    // ...
} MYDATA;

// ...

```

ОПЕРАТОРЫ ПРОГРАММЫ

- Оператор *# include* обеспечивает преимущества использования заголовочных файлов, которые содержат операторы C++ или программные определения.
- Основная часть программы на C++ начинается с оператора *void main(void)*.
- Программы состоят из одной или нескольких функций, которые, в свою очередь, состоят из операторов, предназначенных для решения определенной задачи.
- При выводе на экран ваши программы будут широко использовать выходной поток *cout*.

ОПЕРАТОРЫ ПРОГРАММЫ

- `#include <iostream.h>`
- `void main(void)`
- `{`
 `cout << "Учимся программировать на языке C++!";`
 } // оператор void main(void) указывает начальные (или главные) операторы программы — часть программы, которая выполняется первой.

ОПЕРАТОРЫ ПРОГРАММЫ

- `void main (void) //— ----->` *Программа не возвращает значение*
- `void main (void) //----->` *Программа не использует аргументы командной строки*
- `#include <iostream.h>`
- `void main(void)`
- `{
 cout << 1001;
}`

ОПЕРАТОРЫ ПРОГРАММЫ

- `#include <iostream.h>`
- `void main(void)`
- ```
(
 cout << 1 << 0 << 0 << 1;
)
```
- Если вы не выводите символьную строку, можете поместить символ новой строки внутри одинарных кавычек. Например, следующая программа NEWLINES.CPP выводит числа 1, 0, 0 и 1, каждое на своей собственной строке:
- `#include <iostream.h>`
- `void main(void)`

```
{
 cout << 1 << '\n' << 0 << '\n' << 0 << '\n' << 1;
}
```

# ОПЕРАТОРЫ ПРОГРАММЫ

- адрес издательства "Jamsa Press" в несколько строк:
- `#include <iostream.h>`
- `void main(void)`
- ```
{  
    cout << "Jamsa Press" << endl;  
    cout << "2975 South Rainbow, Suite I" << endl;  
    cout << "Las Vegas, NV 89102" << endl;  
}
```

ВЫВОД ВОСЬМЕРИЧНЫХ И ШЕСТНАДЦАТЕРИЧНЫХ ЗНАЧЕНИЙ

- `#include <iostream.h>`
- `void main(void)`
- `{`
 - `cout << "Восьмеричный: " << oct << 10 << ' ' << 20 << endl;`
 - `cout << "Шестнадцатеричный: " << hex << 10 << ' ' << 20 << endl;`
 - `cout << "Десятичный: " << dec << 10 << ' ' << 20 << endl;`
- `}`

- `#include <iostream.h>`
- `void main(void)`
- `{`
 - `int age ;`
 - `float salary;`
 - `long distance_to_the_moon;`
- `age = 32;`
- `salary = 25000.75;`
- `distance_to_the_moon = 238857;`
- `}`

- Основные математические операции C++
- Операция Назначение Пример + Сложение `total = cost + tax;` - Вычитание `change = payment - total;` * Умножение `tax = cost * tax_rate;` / Деление `average = total / count;`
- `#include <iostream.h>`
- `void main(void)`
- ```
{
 cout << "5 + 7 = " << 5 + 7 << endl;
 cout << "12 - 7 = " << 12 - 7 << endl;
 cout << "1.2345 * 2 = " << 1.23.45 * 2 << endl;
 cout << "15 / 3 = " << 15 / 3 << endl;
}
```

# использование префиксной и постфиксной операций

## увеличения:

- `#include <iostream.h>`
- `void main(void)`
- ```
{
    int small_count = 0;
    int big_count = 1000;
    cout << "small_count равно " << small_count << endl;
    cout << "small_count++ производит " << small_count++ << endl;
    cout << "конечное значение small_count равно " << sniall_count << endl;
    cout << "big_count равно " << big_count << endl;
    cout << "++big_count производит " << ++big_count << endl;
    cout << "конечное значение big_count равно " << big_count << endl;
}
```
- **small_count равно 0**
- **small_count++ производит 0**
- **конечное значение small_count равно 1**
- **big_count равно 1000**
- **++big_count производит 1001**
- **конечное значение big_count равно 1001**

ОПЕРАТОР *else*

- `if (условие_истинно)`
 оператор;
`else`
 оператор;
- `#include <iostream.h>`
- `void main(void)`
- `{`
 `int test_score = 95;`
 `if (test_score >= 90)`
 `cout << "Поздравляю, вы получили A!" << endl;`
 `else`
 `cout << "В следующий раз вы должны" << "`
 `работать усерднее!" << endl;`
 `}`

ОПЕРАТОР *else*

- `#include <iostream.h>`
- `void main(void)`
- ```
{
 int test_score = 65;
 if (test_score >= 90)
 {
 cout << " Поздравляю, вы получили A!" << endl;
 cout << "Ваши тестовые очки были " << test_score << endl;
 }
 else
 {
 cout << "Вы должны работать усерднее!" << endl;
 cout << "Вы потеряли " << 100 - test_score << " очков " <<
endl;
 }
}
```

# ПРОВЕРКА ДВУХ ИЛИ БОЛЕЕ УСЛОВИЙ

- `if ((user_owns_a_dog) && (dog == dalmatian)) //----->Полное условие`
- `if ((user_owns_a_dog) && (dog == dalmatian))`
- `if ((user_owns_a_dog) || (user_owns_a_cat))`
- `#include <iostream.h>`
- `void main(void)`
- `{`
  - `int user_owns_a_dog = 1;`
  - `int user_owns_a_cat = 0;`
  - `if (user_owns_a_dog)`
    - `cout << "Собаки великолепны" << endl;`
  - `if (user_owns_a_cat)`
    - `cout << "Кошки великолепны" << endl;`
  - `if ((user_owns_a_dog) && (user_owns_a_cat))`
    - `cout << "Собаки и кошки могут ужиться" << endl;`
  - `if {(user_owns_a_dog) || (user_owns_a_cat)}`
    - `cout << "Домашние животные великолепны!" << endl;`
- `}`

- `#include <iostream.h>`
- `void main(void)`
- ```
{
    int count;
    int ending_value;
    cout << "Введите конечное значение и
нажмите Enter: ";
    cin >> ending_value;
    for (count = 0; count <= ending_value;
count++)
        cout << count << ' ';
}
```

- `#include <iostream.h>`
- `void main(void)`
- ```
{
 char letter;
 float value;
 for (letter = 'A'; letter <= 'Я'; letter++)
 cout << letter;
 cout << endl;
 for (value = 0.0; value <= 1.0; value += 0.1)
 cout << value << ' ';
 cout << endl;
}
```

- Если вы откомпилируете и запустите эту программу, на экране появится следующий вывод:
- **АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ**
- **0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9**

- `#include <iostream.h>`
- `void main(void)`
- `{`  
`int done = 0; // Устанавливается в состояние „истина“, если введены Д`  
`или H char letter;`  
`while (! done)`
- `{`  
`cout << "\nВведите Д или H" << " и нажмите Enter для продолжения:`  
`”;`  
`cin >> letter;`  
`if ((letter == 'Д') || (letter == 'д'))`  
`done = 1;`  
`else if ((letter == 'H' ) || (letter == 'h'))`  
`done = 1;`  
`else cout << '\a'; // Играть сигнал динамика для неверного`  
`символа`  
`}`  
`cout << "Вы ввели букву " << letter << endl;`  
`}`



# ПРОГРАММА МОЖЕТ ПЕРЕДАВАТЬ ИНФОРМАЦИЮ В ФУНКЦИИ

- `#include <iostream.h>`
- `void show_title (void)`
- `{`  
    `cout << "Книга: Учимся программировать на С++" << endl;`  
`}`
- `void show_lesson (void)`  
`{`  
    `cout << "Урок: Знакомство с функциями" << endl;`  
`}`
- `void main (void)`  
`{`  
    `show_title ();`  
    `show_lesson ();`  
`}`

- C++ будет подставлять переданное число вместо каждого имени параметра `value` внутри функции:
- `show_number (1001)`
- `void show_number (int value )`
- ```
{  
    cout << "Значение параметра равно " << value <<  
endl;  
}
```
- `void show_number (1001)`
- ```
{
 cout << "Значение параметра равно " << 1001 <<
endl;
}
```

- Для каждого передаваемого параметра функция должна указать имя и тип. Например, следующая программа BIGSMALL.CPP использует функцию *show\_big\_and\_little* для вывода самого большого и самого маленького из трех полученных целочисленных значений:
- `#include <iostream.h>`
- `void show_big_and_little (int a, int b, int c)`
- `{`
- `int small = a;`
- `int big = a;`
- `if (b > big)`
- `big = b;`
- `if (b < small)`
- `small = b;`
- `if (c > big)`
- `big = c;`
- `if (c < small)`
- `small = c;`
- `cout << "Самое большое значение равно " << big << endl;`
- `cout << "Самое маленькое значение равно " << small << endl;`
- `}`
- `void main (void)`
- `{`
- `show_big_and_little (1, 2, 3);`
- `show_big_and_little (500, 0, -500);`
- `show_big_and_little (1001, 1001, 1001);`
- `}`