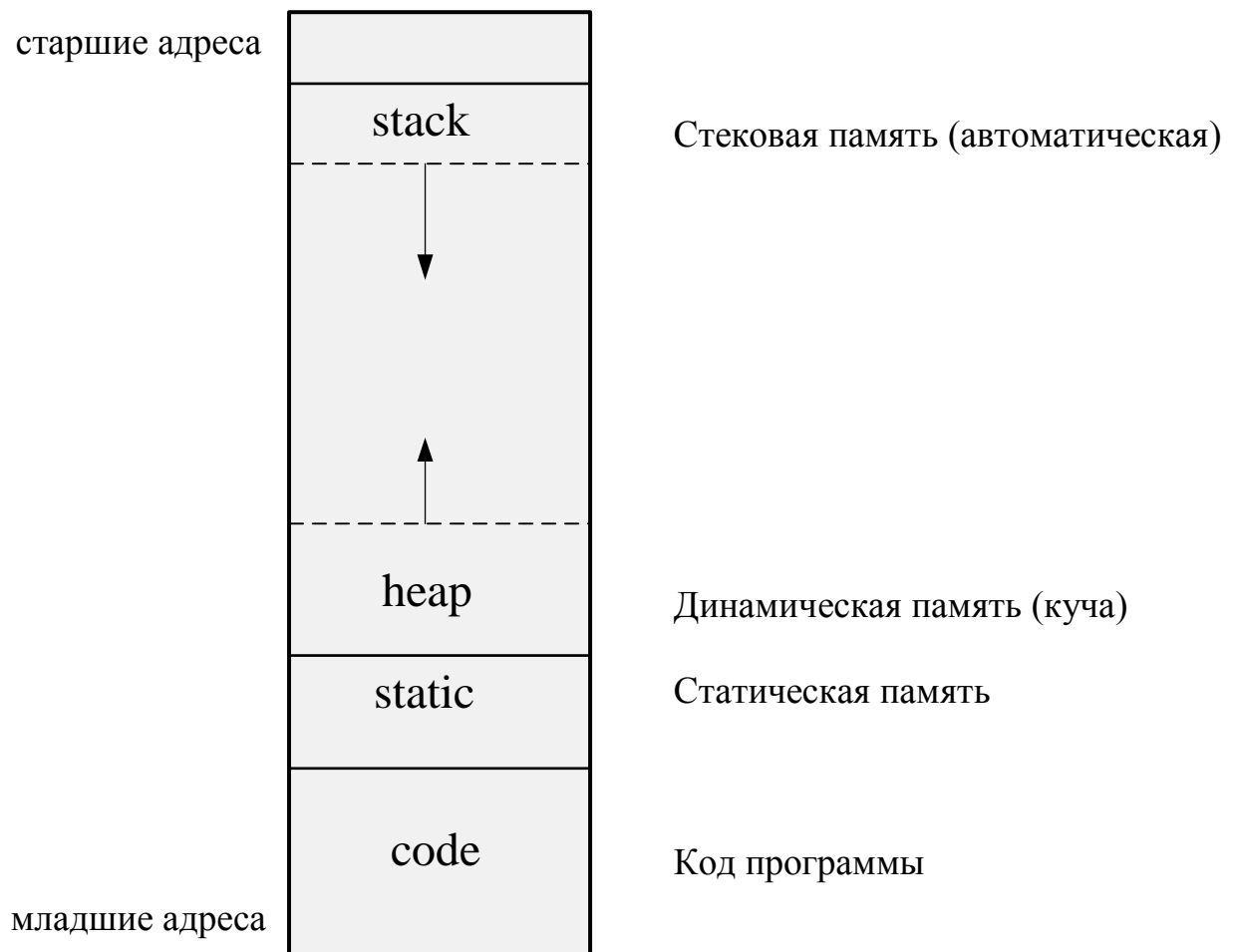


Структура языка программирования

1. Классы памяти C++:

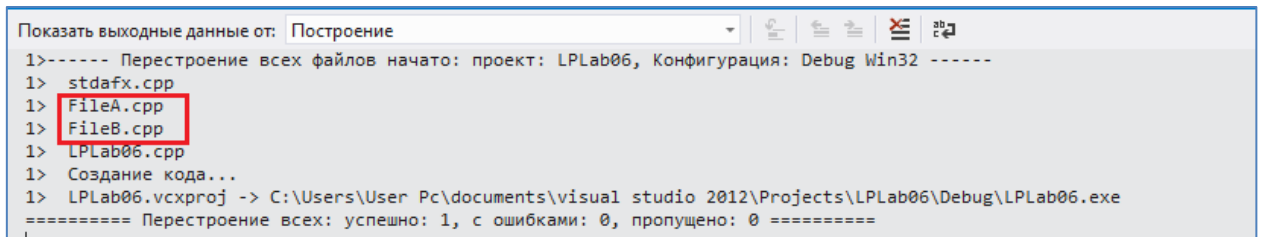
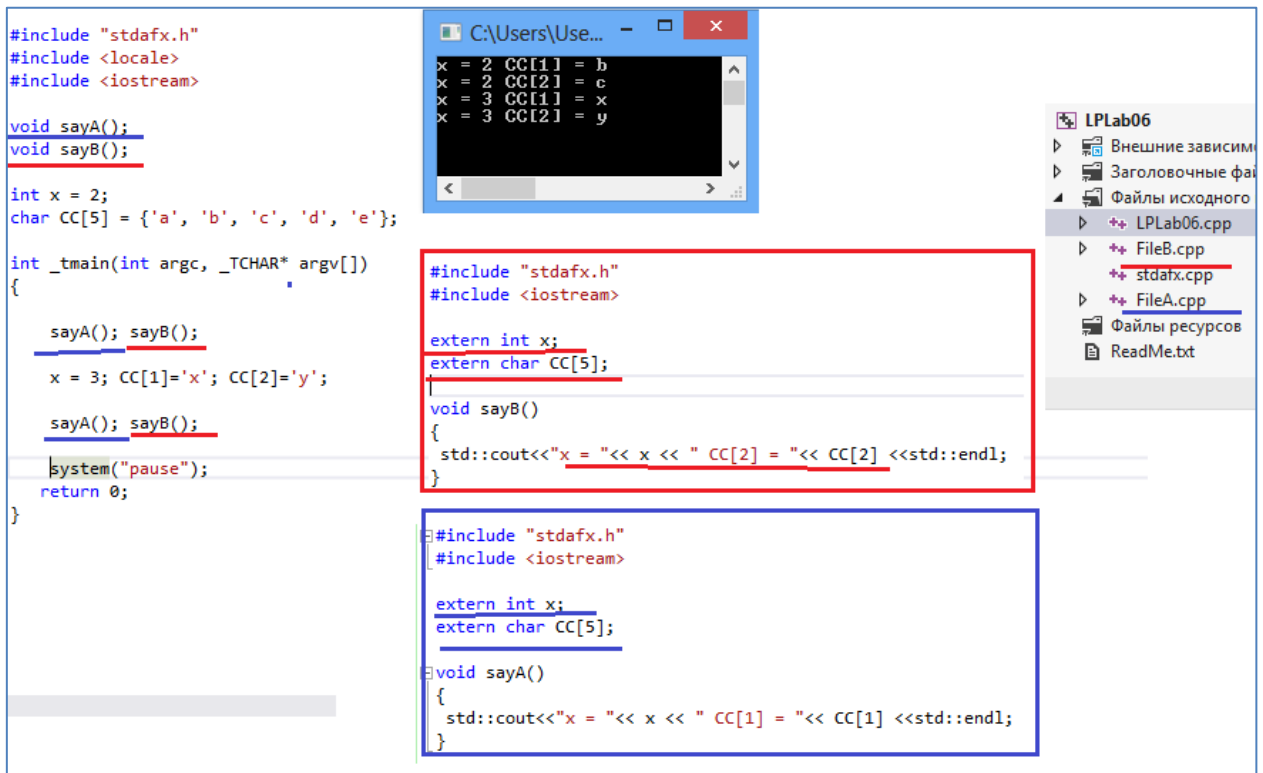
код;
статические данные (глобальные);
стек (локальные переменные, параметры);
куча (динамическая память).

Структура памяти C-программ:



2. Классы памяти C++:

объявление глобальных статических переменных с ключевым словом **extern** (компонуются редактором связей (linker)).



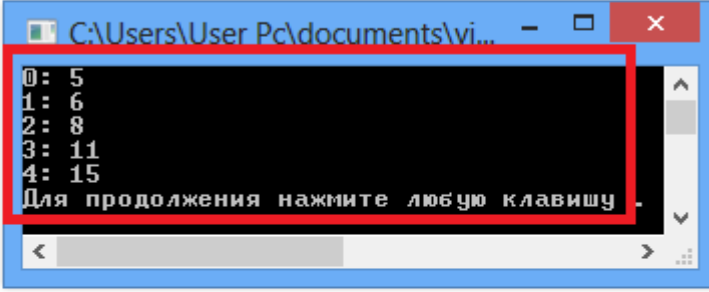
3. Классы памяти C++:

локальная статическая память.

```
#include "stdafx.h"
#include <locale>
#include <iostream>

int func (int x)
{
    static int k = 5;
    return k+=x;
};

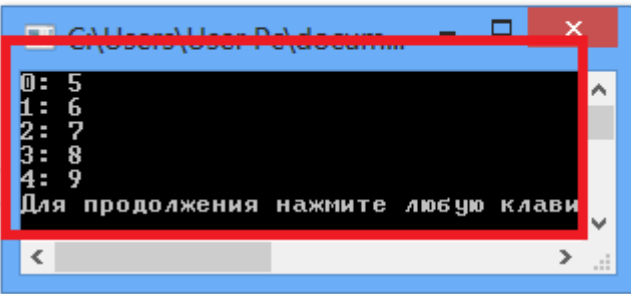
int _tmain(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 5; i++)
    {
        std::cout << i << ": " << func(i) << std::endl;
    };
    system("pause");
    return 0;
}
```



```
#include "stdafx.h"
#include <locale>
#include <iostream>

int func (int x)
{
    int k = 5;
    return k+=x;
};

int _tmain(int argc, _TCHAR* argv[])
{
    for (int i = 0; i < 5; i++)
    {
        std::cout << i << ": " << func(i) << std::endl;
    };
    system("pause");
    return 0;
}
```



4. Классы памяти C++:

стек

```
#include "stdafx.h"
#include <locale>
#include <iostream>

int func2 (int y, int z)
{
    int g = 8;
    int h = 9;
    int r = 10;
    return y + z + g + h + r;
};

int func1 (int x, int v)
{
    int k = 5;
    int l = 6;
    int m = 7;
    return k+= func2(k, 7);
};

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << func1(3, 4)<< std::endl;
    return 0;
}
```

argc

argv

```
#include "stdafx.h"
#include <locale>
#include <iostream>

int func2 (int y, int z)
{
    int g = 8;
    int h = 9;
    int r = 10;
    return y + z + g + h + r;
};

int func1 (int x, int v)
{
    int k = 5;
    int l = 6;
    int m = 7;
    return k+= func2(k, 7);
};

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << func1(3, 4)<< std::endl;
    return 0;
}
```

m

l

k

x

v

argc

argv

```

#include "stdafx.h"
#include <locale>
#include <iostream>

int func2 (int y, int z)
{
    int g = 8;
    int h = 9;
    int r = 10;
    return y + z + g + h + r;
};

int func1 (int x, int v)
{
    int k = 5;
    int l = 6;
    int m = 7;
    return k+= func2(k, 7);
};

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << func1(3, 4)<< std::endl;
    return 0;
}

```

r
h
g
y
z
m
l
k
x
v
argc
argv

```

#include "stdafx.h"
#include <locale>
#include <iostream>

int func2 (int y, int z)
{
    int g = 8;
    int h = 9;
    int r = 10;
    return y + z + g + h + r;
};

int func1 (int x, int v)
{
    int k = 5;
    int l = 6;
    int m = 7;
    return k+= func2(k, 7);
};

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << func1(3, 4)<< std::endl;
    return 0;
}

```

m
l
k
x
v
argc
argv

```

#include "stdafx.h"
#include <locale>
#include <iostream>

int func2 (int y, int z)
{
    int g = 8;
    int h = 9;
    int r = 10;
    return y + z + g + h + r;
};

int func1 (int x, int v)
{
    int k = 5;
    int l = 6;
    int m = 7;
    return k+= func2(k, 7);
};

int _tmain(int argc, _TCHAR* argv[])
{
    std::cout << func1(3, 4)<< std::endl;
    return 0;
}

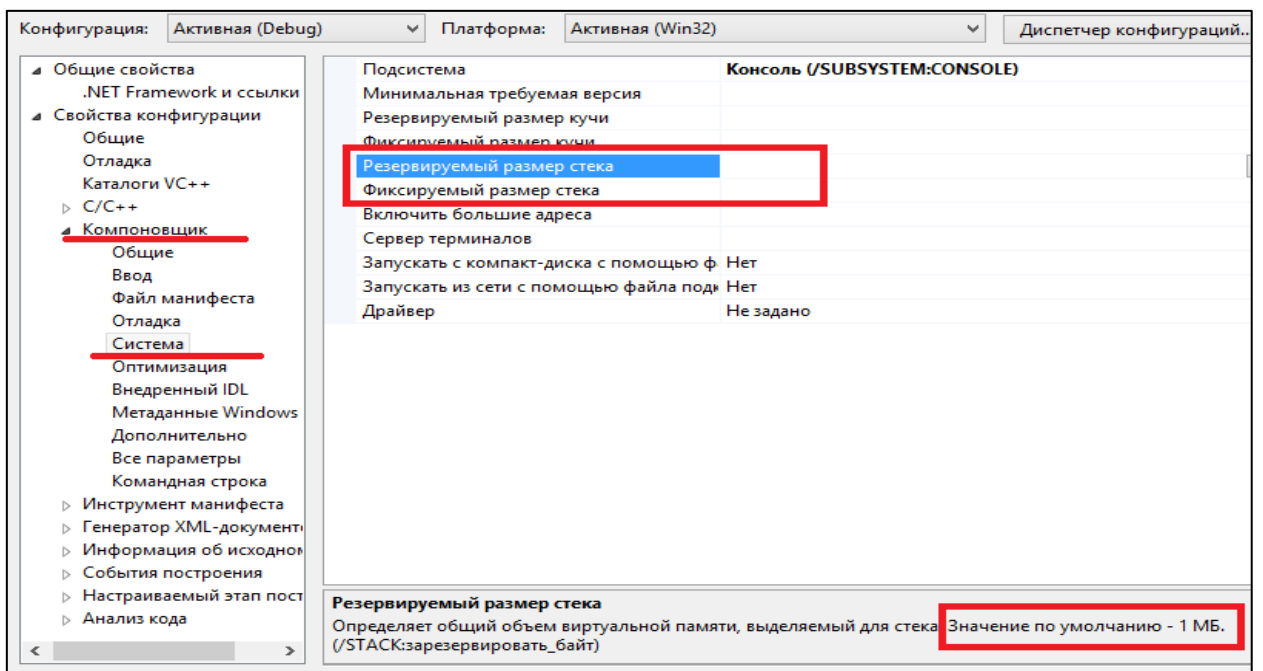
```

argc

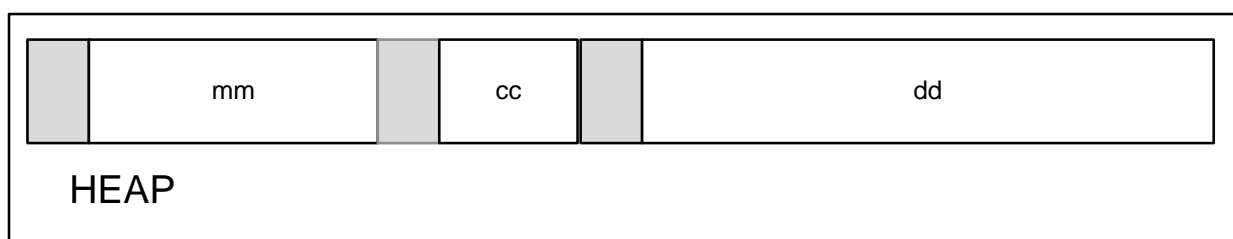
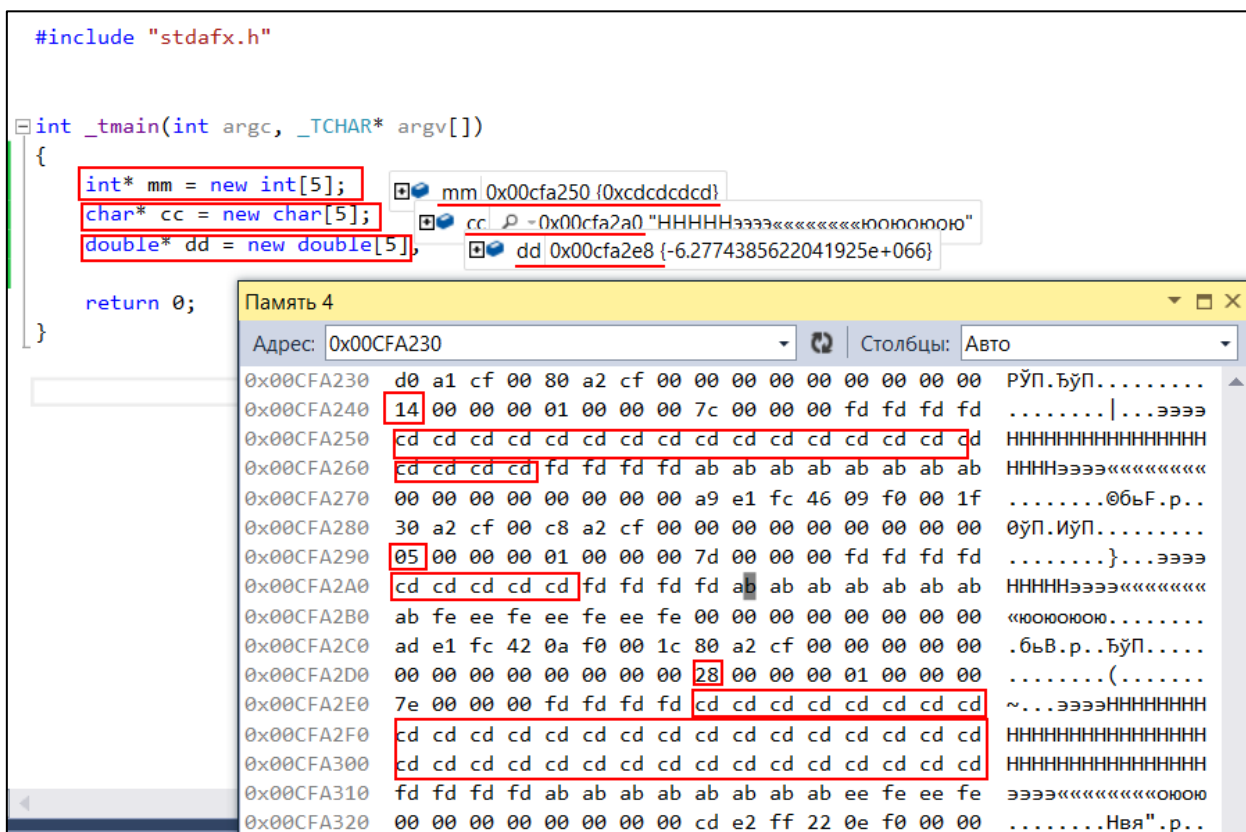
argv

5. Классы памяти C++:

СТЕК



hear (куча).



7. Обработка ошибок:

механизм исключений.

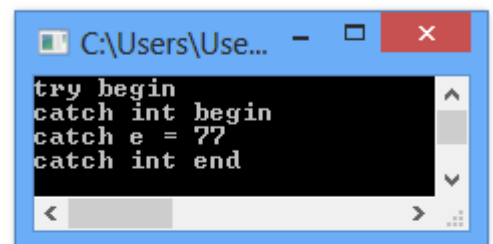
Исключение – событие при выполнении программы, при котором ее дальнейшее выполнение становится бессмысленным.

8. Обработка ошибок:

механизм исключений (try/catch/throw)

```
struct EEE {int k; char c;};

int _tmain(int argc, _TCHAR* argv[])
{
    try
    {
        std::cout << "try begin"<< std::endl;
        throw (int)77;
        std::cout << "try end" << std::endl;
    }
    catch (int e)
    {
        std::cout << "catch int begin"<< std::endl;
        std::cout << "catch e = "<<e<<std::endl;
        std::cout << "catch int end" << std::endl;
    }
    catch(char e)
    {
        std::cout << "catch char begin"<< std::endl;
        std::cout << "catch e = "<<e<<std::endl;
        std::cout << "catch char end" << std::endl;
    }
    catch(EEE e)
    {
        std::cout << "catch EEE begin"<< std::endl;
        std::cout << "catch e = "<<e.c << e.k<<std::endl;
        std::cout << "catch EEE end" << std::endl;
    }
}
```



```
C:\Users\Use... - [X]
try begin
catch int begin
catch e = 77
catch int end
```



```

int _tmain(int argc, _TCHAR* argv[])
{
    try
    {
        std::cout << "try begin"<< std::endl;
        EEE eee = {25, 'M'};
        throw eee;
        std::cout << "try end" << std::endl;
    }
    catch (int e)
    {
        std::cout << "catch int begin"<< std::endl;
        std::cout << "catch e = "<<e<<std::endl;
        std::cout << "catch int end" << std::endl;
    }
    catch(char e)
    {
        std::cout << "catch char begin"<< std::endl;
        std::cout << "catch e = "<<e<<std::endl;
        std::cout << "catch char end" << std::endl;
    }
    catch(EEE e)
    {
        std::cout << "catch EEE begin"<< std::endl;
        std::cout << "catch e = "<<e.c << e.k<<std::endl;
        std::cout << "catch EEE end" << std::endl;
    }
}

```

```

C:\Users\... - [X]
try begin
catch EEE begin
catch e = M25
catch EEE end

```

```

#include "stdafx.h"
#include <locale>
#include <iostream>

void func();

int _tmain(int argc, _TCHAR* argv[])
{
    try
    {
        std::cout << "main try begin"<<std::endl;
        func();
        std::cout << "main try end"<<std::endl;
    }
    catch(int e)
    {
        std::cout << "main catch int:"<< e<<std::endl;
    }
    catch(char* e)
    {
        std::cout << "main catch char*:"<< e<<std::endl;
    }
    system("pause");
    return 0;
}

```

```

C:\Users\User Pc\documents\visual stud... - [X]
main try begin
func try begin
main catch char*:func throw
Для продолжения нажмите любую клавишу . . .

```

```

#include "stdafx.h"
#include <locale>
#include <iostream>

void func()
{
    try
    {
        std::cout << "func try begin"<<std::endl;
        throw "func throw";
        std::cout << "func func try end"<<std::endl;
    }
    catch(int e)
    {
        std::cout << "func catch int:"<< e<<std::endl;
    }
}

```

9. Обработка ошибок:

Пример:

```
#include "stdafx.h"
#include <locale>
#include <iostream>
namespace Date
{
    // количество дней между датами
    unsigned long distance(short yyyy1, short mm1, short dd1,
                           short yyyy2, short mm2, short dd2);
};

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "rus");
    try
    {
        // Джон фон Нейман (28.12.1903 - 9.02.1957)
        long d1 = Date::distance(1957, 2, 9, 1903, 12, 28);
        std::cout<<" Джон фон Нейман прожил "<<d1<<" дней"<<std::endl;
        // от рождества Христова.
        long d2 = Date::distance(0, 1, 7, 2015, 3, 24);
        std::cout<<"От рождества Христова прошло "<<d2<<" дней"<<std::endl;
        // Ада Лавлейс (10.12.1815 - 27.12.1852)
        long d3 = Date::distance(1852, 12, 27, 2015, 3, 24);
        std::cout<<"Со дня смерти Ады Лавлейс прошло "<<d3<<" дней"<<std::endl;
    }
    catch (char* e) { std::cout<<" Ошибка: "<<e <<std::endl; }

    system("pause");
    return 0;
}
```

Джон фон Нейман прожил 19402 дней
Ошибка: Date: год должен быть 1 или больше

```
#include "stdafx.h"
#include <locale>
#include <iostream>
namespace Date
{
    unsigned long datetoday(short yyyy, short mm, short dd)
    {
        bool G = (yyyy < 1582) || (yyyy == 1582 && mm < 10) || (yyyy == 1582 && mm == 10 && dd < 15);
        //int A = (G?0:2-(yyyy/100) + (yyyy/400)); // это правильно
        int A = 2-(yyyy/100) + (yyyy/400); // так у Microsoft
        mm = (mm <= 2? (yyyy--, mm+12): mm);
        unsigned long rc = (1461L * long(yyyy))/4L;
        unsigned long k = (306001L * long(mm+1))/10000L;
        rc += k + dd + 1720995L + A;
        return rc;
    };

    unsigned long distance(short yyyy1, short mm1, short dd1, short yyyy2, short mm2, short dd2)
    {
        if (yyyy1 < 1 || yyyy2 < 1) throw "Date: год должен быть 1 или больше";
        if (mm1 < 1 || mm1 > 12 || mm2 < 1 || mm2 > 12) throw "Date: месяц должен быть в интервале от 1 до 12";
        if (dd1 < 1 || mm1 > 31 || dd2 < 1 || dd2 > 31) throw "Date: день должен быть в интервале от 1 до 31";
        if (dd1 > 28 && yyyy1%4 > 0) throw "Date: день должен быть в интервале от 1 до 28";
        if (dd1 > 29 && yyyy1%4 == 0) throw "Date: день должен быть в интервале от 1 до 29";
        return datetoday(yyyy1, mm1, dd1) - datetoday(yyyy2, mm2, dd2);
    };
}
```

10. Обработка ошибок:

в стандартной библиотеке определено несколько стандартных типов исключений.

11. Инструкции языка:

- инструкции объявления (if (int) условное объявление);
- составные инструкции ({});
- инструкции выбора (if, switch);
- циклы (while, do-while, for);
- инструкции перехода (goto, break, continue, return);
- обработка исключений (try, catch, throw).