

# Программирование на языке C++

## Лекция 2

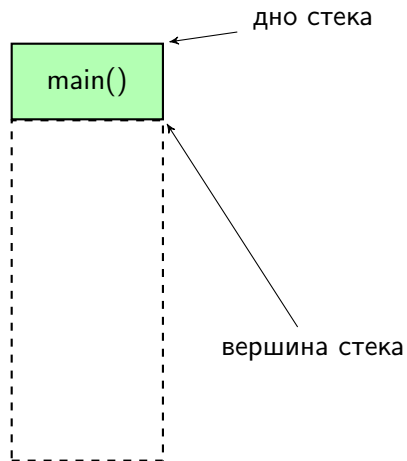
Стек вызовов

Александр Смаль

# Стек вызовов

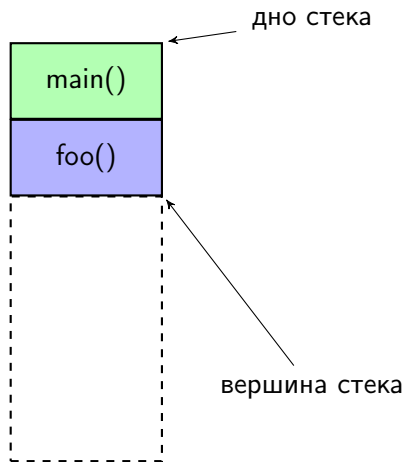
- Стек вызовов — это сегмент данных, используемый для хранения локальных переменных и временных значений.
- Не стоит путать стек с одноимённой структурой данных, у стека в C++ можно обратиться к произвольной ячейке.
- Стек выделяется при запуске программы.
- Стек обычно небольшой по размеру (4Мб).
- Функции хранят свои локальные переменные на стеке.
- При выходе из функции соответствующая область стека объявляется свободной.
- Промежуточные значения, возникающие при вычислении сложных выражений, также хранятся на стеке.

# Устройство стека



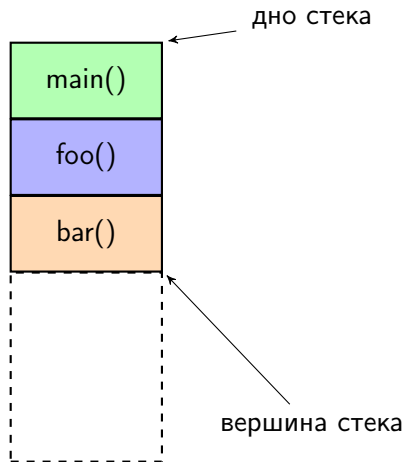
```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```

# Устройство стека



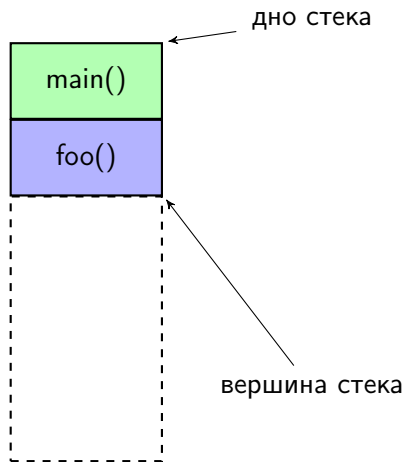
```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```

# Устройство стека



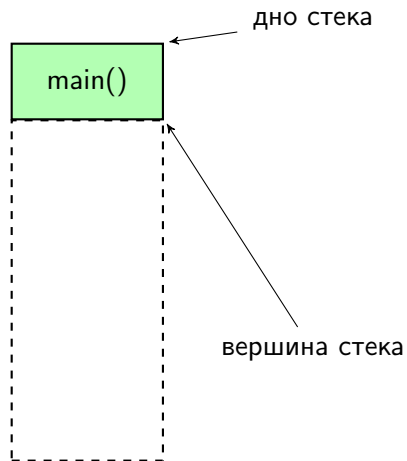
```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```

# Устройство стека



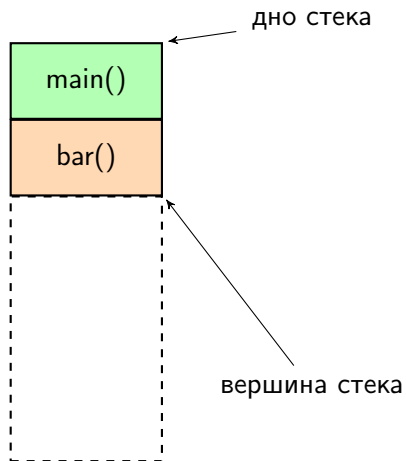
```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```

# Устройство стека



```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```

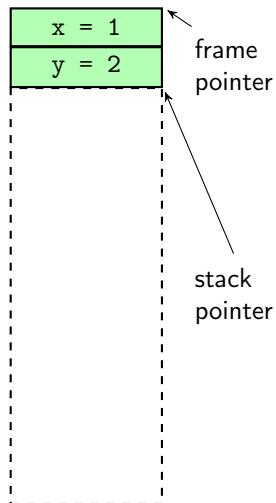
# Устройство стека



```
void bar( ) {  
    int c;  
}  
  
void foo( ) {  
    int b = 3;  
    bar();  
}  
  
int main( ) {  
    int a = 3;  
    foo();  
    bar();  
  
    return 0;  
}
```



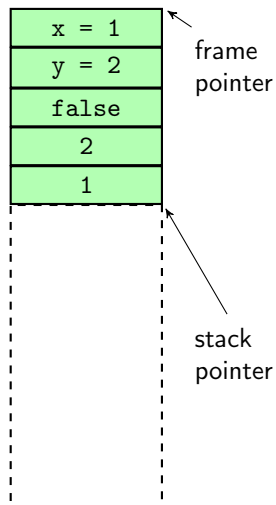
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

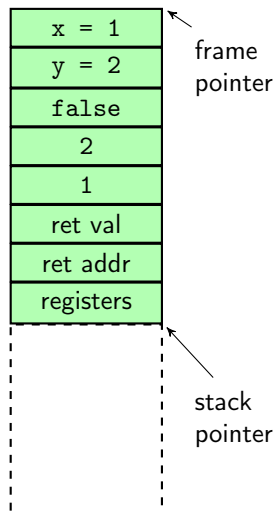
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

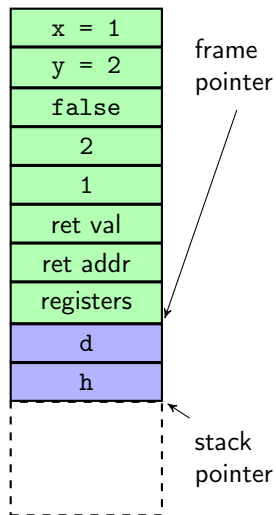
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

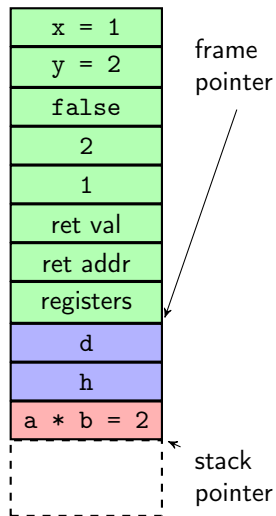
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

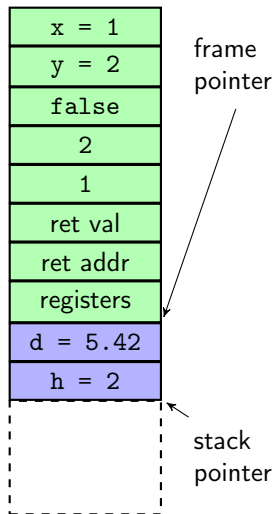
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

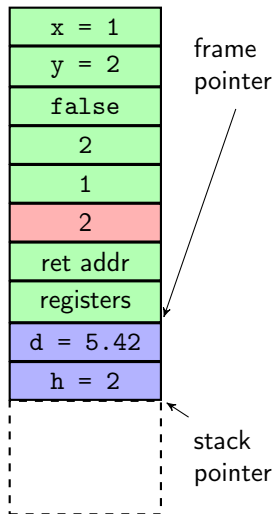
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

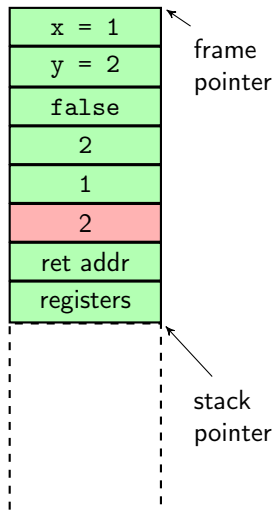
# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

# Вызов функции

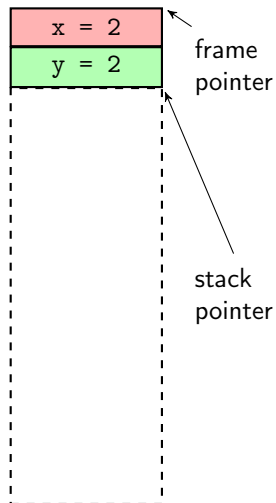


```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```



# Вызов функции



```
int foo(int a, int b, bool c)
{
    double d = a * b * 2.71;
    int h = c ? d : d / 2;
    return h;
}

int main( )
{
    int x = 1;
    int y = 2;
    x = foo (x, y, false);
    cout << x;
    return 0;
}
```

# Вызов функции

- При вызове функции на стек складываются:
  1. аргументы функции,
  2. адрес возврата,
  3. значение frame pointer и регистров процессора.
- Кроме этого на стеке резервируется место под возвращаемое значение.
- Параметры передаются в обратном порядке, что позволяет реализовать функции с переменным числом аргументов.
- Адресация локальных переменных функции и аргументов функции происходит относительно frame pointer.
- Конкретный процесс вызова зависит от используемых соглашений (cdecl, stdcall, fastcall, thiscall).