Оптимизации времени компиляции

DINS®

Обо мне

Embedded microcontroller programming

Embedded programming

Dino systems

О чем доклад

Не серебряная пуля

Передача информации или метаинформации на этапе компиляции
От места объявления к месту использования

Что не попадет

Defines

Const

Noexcept

Что попадет

Факториалы

Грека

Бессмысленные оптимизации

Виды оптимизации

Оптимизация - инлайнинг

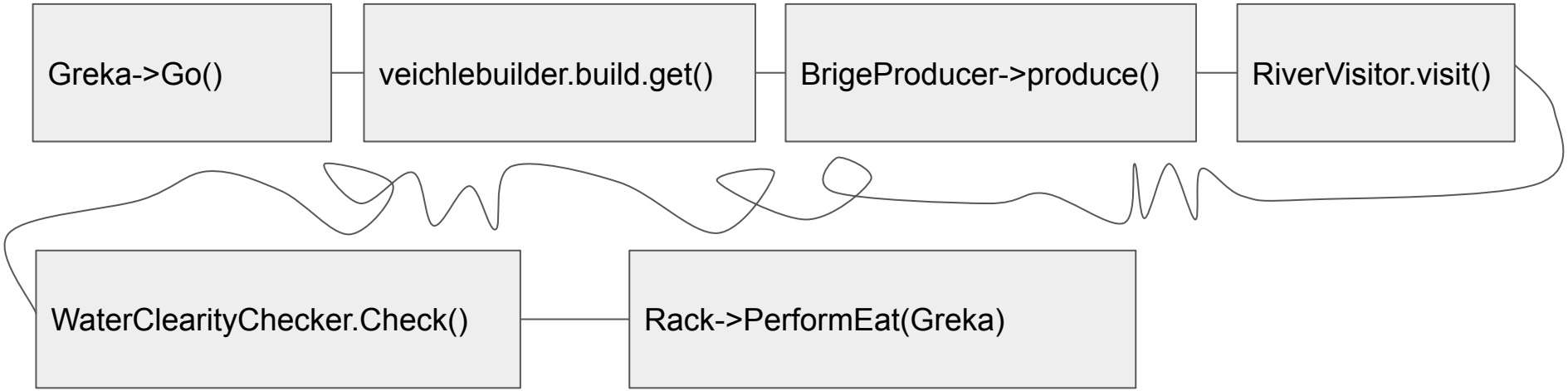
```
int addThree(int a)
{return a+3;};

int addFour(int a)
{return a+4;};

int addFive(int a)
{return a+5;};

int Calc2+3+4+5() {
return addFive(addFour(addThree(2)));
}
```

Оптимизация - инлайнинг

```
int addThree(int a)
{return a+3;};

int addFour(int a)
{return a+4;};

int addFive(int a)
{return a+5;};

int Calc2+3+4+5() {
return addFive(addFour(addThree(2)));
}
```

```
addThree(int):
lea eax, [rdi+3]
ret
addFour(int):
lea eax, [rdi+4]
ret
addFive(int):
lea eax, [rdi+5]
ret
Calc2345():
mov eax, 14
ret
```

Оптимизация - инлайнинг



Оптимизация - выкидывание ненужного

```
#include "array"
const std::array<int,10> arr{1,2,3,4,5,6,7,8,9,10};

int returnSome()
{
return arr[5];
}
```

Оптимизация - выкидывание ненужного

```
#include "array"
const std::array arr{1,2,3,4,5,6,7,8,9,10};

int returnSome()
{
return arr[5];
}
```

Оптимизация - выкидывание ненужного

```
#include "array"
const std::array arr{1,2,3,4,5,6,7,8,9,10};

int returnSome()
{
return arr[5];
}
```

```
returnSome():
mov eax, 6
ret
```

Компилятор

Frontend

Middleend

Backend

Расчет на фронтенде

```
 int factorial (int n)
{
   return n > 0 ? n * factorial( n - 1 ) : 1;
}


int main()
{
   return factorial(10);
}
```

Расчет на фронтенде -O2

```
 int factorial (int n)
{
   return n > 0 ? n * factorial( n - 1 ) : 1;
}


int main()
{
   return factorial(10);
}
```

```
main:
mov eax, 1
mov edx, 10
.L8:
imul eax, edx
sub edx, 1
jne .L8
ret
```

Расчет на фронтенде -O3

```
 int factorial (int n)
{
   return n > 0 ? n * factorial( n - 1 ) : 1;
}


int main()
{
   return factorial(10);
}
```

```
main:
mov eax, 3628800
ret
```

Расчет на фронтенде

```
 constexpr int factorial (int n)
{
   return n > 0 ? n * factorial( n - 1 ) : 1;
}


int main()
{
   return factorial(10);
}
```

```
main:
mov eax, 3628800
ret
```

Linker

Text

Data

Bss

Статическая инициализация

Статическая инициализация

```
int factorial (int n)
{
return n > 0 ? n * factorial( n - 1 ) : 1;
}
const int data = factorial(10);
int main()
{
return data;
}
```

```
main:
mov eax, DWORD PTR data[rip]
ret

_GLOBAL__sub_I_factorial(int):
mov DWORD PTR data[rip], 3628800
ret
```

Статическая инициализация

```
constexpr int factorial (int n)
{
return n > 0 ? n * factorial( n - 1 ) : 1;
}
const int data = factorial(10);
int main()
{
return data;
}
```

```
main:
mov eax, 3628800
ret
```

Статическая инициализация

```
constexpr int factorial (int n)
{
return n > 0 ? n * factorial( n - 1 ) : 1;
}
constexpr int data = factorial(10);
int main()
{
return data;
}
```

```
main:
mov eax, 3628800
ret
```

Static (local) const

```
int mainWithStaticData()

{
static const std::array<int,2> a = {1,2};
return a[1];
}
```

```
mainWithStaticData():
mov eax, 2
ret
```

Static (local) const

```
int mainWithStaticData()

{
static const std::vector<int> a {3,2,1,0};
return a[1];
}
```

```
mainWithStaticData():
movzx eax, BYTE PTR guard variable for mainWithStaticData()::a[rip]
test al, al
je .L20
mov rax, QWORD PTR mainWithStaticData()::a[rip]
mov eax, DWORD PTR [rax+4]
ret
.L20:
push rbp
mov edi, OFFSET FLAT:guard variable for mainWithStaticData()::a
call __cxa_guard_acquire
test eax, eax
jne .L21
mov rax, QWORD PTR mainWithStaticData()::a[rip]
pop rbp
mov eax, DWORD PTR [rax+4]
ret
.L21:
pxor xmm0, xmm0
mov edi, 16
```

Как сделать лучше?

Чтобы добиться максимальной производительности, нужно дать компилятору максимальную информацию об используемых переменных

Метаданные
Сами переменные (данные)
Места использования
Способы использования

Что сейчас оптимизацией на этапе компиляции

Или ничего или...

Если много времени

protobuff

```
template<typename A, typename B>

struct and_
: public integral_constant<bool, (A::value && B::value)>
{};
```

Если мало времени

LTO

namespace{}

const

static

Но есть же constexpr...

Но есть же constexpr...

C++11

```
constexpr void LessThan10(int a,bool check)

{
if (check)
assert(a < 10);
}
```

Но есть же constexpr...

C++11

```
constexpr void LessThan10(int a,bool check)

{
if (check)
assert(a < 10);
}
```

Но есть же constexpr...

C++11

```
constexpr void LessThan10(int a,bool check)

{
if (check)
assert(a < 10);
}

reinterpret_cast
```

Но есть же constexpr...

Constexpr lambda
Constexpr if
Constexpr allocator
Consteval
Constexpr virtual destructor

Если вообще не оптимизировать -O2

```
unsigned int powerOfFive(unsigned int i)
{
if (i==0) return 1;
return powerOfFive(i-1)*5;
}
int calc()
{
return powerOfFive(2) * powerOfFive(8);
}

void BusinessLogic()
{
assert(calc() > 155);
}
```

```
BusinessLogic():
ret
```

Если вообще не оптимизировать -O2

```
unsigned int powerOfFive(unsigned int i)
{
if (i==0) return 1;
return powerOfFive(i-1)*5;
}
int calc()
{
return powerOfFive(2) * powerOfFive(9);
}

void BusinessLogic()
{
assert(calc() > 155);
}
```

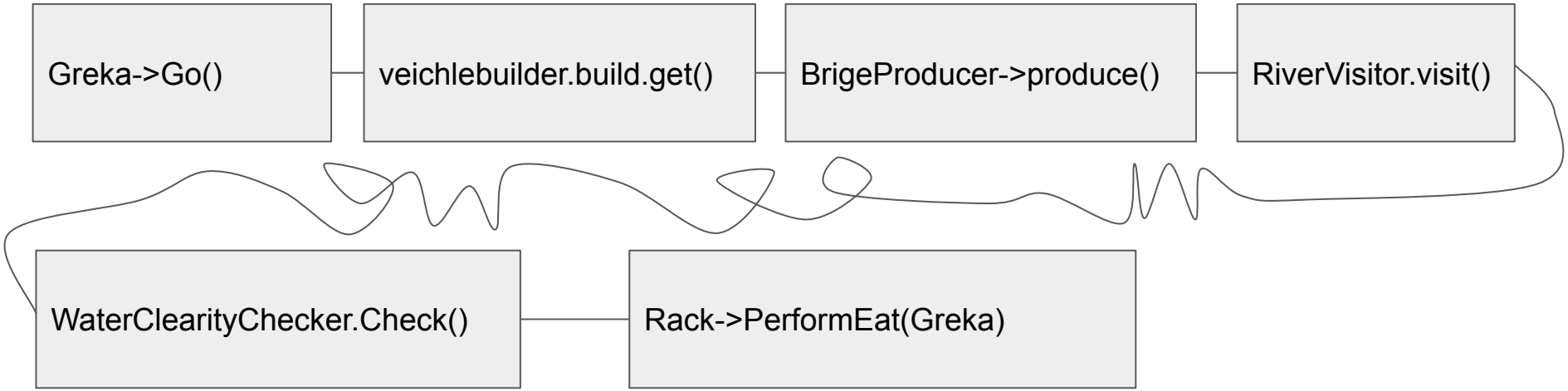
Если вообще не оптимизировать -O2

```
unsigned int powerOfFive(unsigned int i)
{
if (i==0) return 1;
return powerOfFive(i-1)*5;
}
int calc()
{
return powerOfFive(2) * powerOfFive(9);
}

void BusinessLogic()
{
assert(calc() > 155);
}
```

```
BusinessLogic():
mov edx, 9
mov eax, 1
.L15:
mov ecx, eax
lea eax, [rax+rax*4]
sub edx, 1
jne .L15
imul ecx, ecx, 125
cmp ecx, 155
jle .L23
ret
.L23:
push rax
mov ecx, OFFSET FLAT:.LC0
mov edx, 15
mov esi, OFFSET FLAT:.LC1
mov edi, OFFSET FLAT:.LC2
call __assert_fail
```

```
.LC0:
.string "void BusinessLogic()"

.LC1:
.string "/tmp/compiler-explorer-compiler…
.LC2:
.string "calc() > 155"
```

Если вообще не оптимизировать



Если оптимизировать не по уму

```
typedef static_Gpio<Port::A,7,GpioMode::Out,GpioOutType::PP,GpioSpeed::s100MHz,
GpioPuPd::PullDown> led1;
```

```
led1::init();
led1::toggle();
```

Если оптимизировать не по уму

```
typedef DacDelayPinList<
            Port::D,10,
            Port::B,2,
            Port::D,12,
            Port::D,13,
            Port::D,14,
            Port::D,15,
            Port::A,12,
            Port::D,11,
            Port::E,10,
            Port::B,6,
            Port::B,7,
            Port::B,14,
            Port::A,5,
            Port::C,9> dacDelayPinList1;
```

И что нам делать?

Нам нужен метод позволяющий

- обеспечивать максимально возможный уровень оптимизации
- гарантии (упал если в compile time не получилось)
- гибкость (если не нужно в compile time, то не нужно переписывать весь код)
- читаемость

Translation Units

greka.cpp

```
#include “greka.h”
#include “river.h”
```

greka.cpp + greka.h + river.h = translation unit 1

river.cpp + greka.h + river.h = translation unit 2

Translation Units

```
std::array grekasHands {1, 2};


int getGrekasRightHand()
{
    return grekasHands[1];
}
```

```
getGrekasRightHand():
mov eax, DWORD PTR grekasHands[rip+4]
ret
grekasHands:
.long 1
.long 2
```

Translation Units

```
namespace
{
std::array grekasHands {1, 2};
}

int getGrekasRightHand()
{
   return grekasHands[1];
}
```

```
getGrekasRightHand():
mov eax, 2
ret
```

Translation Units

```
static std::array grekasHands {1, 2};


int getGrekasRightHand()
{
    return grekasHands[1];
}
```

```
getGrekasRightHand():
mov eax, 2
ret
```

LTO

greka.cpp

```
#include “greka.h”
#include “river.h”
```

greka.cpp + greka.h + river.h = translation unit 1

river.cpp + greka.h + river.h = translation unit 2

LTO

```
struct Base
{
 virtual int foo() { return 10; };
 virtual ~Base() = default;
};
struct Derived: Base
{
 int foo() override { return 30; };
 virtual ~Derived() = default;
};

Base* createInstance()
{
 return new Derived;
}
```

LTO

```
struct Base
{
 virtual int foo() { return 10; };
 virtual ~Base() = default;
};
struct Derived: Base
{
 int foo() override { return 30; };
 virtual ~Derived() = default;
};

Base* createInstance()
{
 return new Derived;
}
```

```
int main()
{ Base* base = createInstance();
   int result = base->foo();
   delete base;
   return result;}
```

LTO

```
struct Base
{
 virtual int foo() { return 10; };
 virtual ~Base() = default;
};
struct Derived: Base
{
 int foo() override { return 30; };
 virtual ~Derived() = default;
};

Base* createInstance()
{
 return new Derived;
}
```

```
int main()
{ Base* base = createInstance();
   int result = base->foo();
   delete base;
   return result;}
main:
sub rsp, 8
mov edi, 8
call operator new(unsigned long)
mov QWORD PTR [rax], OFFSET FLAT:vtable for Derived+1
mov rdi, rax
call Derived::~Derived() [deleting destructor]
mov eax, 30
add rsp, 8
ret
```

LTO

One.h

```
Base* createInstance();

struct Base
{
 virtual int foo() { return 10; };
 virtual ~Base() = default;
};

struct Derived: Base
{
 int foo() override { return 30; };
 virtual ~Derived() = default;
};
```

main.cpp

```
int main()
{ Base* base = createInstance();
   int result = base->foo();
   delete base;
   return result;}
```

One.cpp

```
Base* createInstance()
{
 return new Derived;
}
```

Single compilation unit

To improve compilation times it is possible to use “unity builds”, called Jumbo builds, in Chromium. The idea is to merge many translation units (“source files”) and compile them together.

Inline переменные

```
struct MyStruct
{
inline static int data = 10;
inline static std::vector<int>
countdown {3,2,1,0};
};

int returnData()
{
return MyStruct::countdown[1];
}
```

```
returnData():
mov rax, QWORD PTR MyStruct::countdown[rip]
mov eax, DWORD PTR [rax+4]
ret
_GLOBAL__sub_I_returnData():
cmp BYTE PTR guard variable for MyStruct::countdown[rip], 0
je .L16
ret
```

loki mpl hana

```
BOOST_HANA_CONSTEXPR_CHECK(
      hana::replace_if(hana::make_tuple(-3, -2, -1, 0, 1, 2, 3), negative, 0)
              ==
      hana::make_tuple(0, 0, 0, 0, 1, 2, 3))
```

Шаблоны передача данных

Шаблоны передача данных

```
static const std::array<int,2> GrekasHands = {1,2};
```

Шаблоны передача данных

```
template <int T>
constexpr int Factorial()
{return Factorial<T-1>() * T;}

template<>
constexpr int Factorial<0>()
{return 0;}

template<>
constexpr int Factorial<1>()
{return 1;}
constexpr int a = Factorial<2>();
```

Шаблоны Статический полиморфизм

Шаблоны Статический полиморфизм

```
template <typename T>
auto performEat(T&& obj)
{
return eat(obj.getHands());
};



performEat(greka);
```

Шаблоны SFINAE

Compile-time switch for templates

Шаблоны SFINAE

```
template<class T>
typename std::enable_if<std::is_floating_point<T>::value, T>::type
 T  foo1(T t)
{   std::cout << "foo1: float\n";
   return t;
}
template<class T>
typename std::enable_if<std::is_integral<T>::value, T>::type
 T  foo1(T t)
{   std::cout << "foo1: int\n";
   return t;
}
```

Шаблоны Концепты

```
template<Integral T>
 T foo1(T t)
{
   std::cout << "foo1: int\n";
   return t;
}
```

Шаблоны Концепты

```
template <class T>
concept Floating = std::is_floating_point<T>;


template<Floating T>
 T foo1(T t)
{
   std::cout << "foo1: float\n";
   return t;
}
```

Шаблоны Концепты

```
template<typename T>
concept Hashable = requires(T a) {
   { std::hash<T>{}(a) } -> std::size_t;
};

template<typename T> requires Hashable<T> && Floating<T>
void f(T);

template<typename T> requires Hashable<T> || Floating<T>
class MySet;
```

Шаблоны Метапрограммирование

```
template <int... Args>
struct SummAllData
  : public std::integral_constant<int, (... + Args)> {};

int main() {
return SummAllData<1,4,3>();
}
```

Шаблоны Метапрограммирование

```
template <typename... Args>
struct AnyHaveBacon
   : public std::integral_constant<bool, (... || Args::bacon)> {};

struct Burger{static const bool hands = true;};
struct Sandwitch{static const bool hands = false;};

std::string kosher(){
if (AnyHaveBacon<Burger, Sandwitch>::value) { return "nope"; }
 else {return "kosher"; }
}
```

Constexpr

```
template<typename... Args>
constexpr int haveBacon(Args... dish) {
   return (dish.bacon || ... );
}
struct Burger{static const bool bacon = true;};
struct Sandwitch{static const bool bacon = false;};

std::string main(){
constexpr Burger burger;
constexpr Sandwitch sandwitch;
if (haveBacon(burger, sandwitch))
{return "nope";}
else{return "kosher";} }
```

Constexpr

```
template<typename... Args>
constexpr int haveBacon(Args... dish) {
   return (dish.bacon || ... );
}
struct Burger{const bool bacon=true;};
struct Sandwitch{const bool bacon=false;};

std::string main(){
constexpr Burger burger;
constexpr Sandwitch sandwitch;
if (haveBacon(burger, sandwitch))
{return "nope";}
else{return "kosher";} }
```

Constexpr

```
template<typename... Args>
constexpr int haveBacon(Args... dish) {
   return (dish.bacon || ... );
}
struct Burger{const bool  bacon=true;};
struct Sandwitch{const bool bacon;
constexpr Sandwitch(bool haveBacon):bacon(haveBacon){};};

std::string main(){
constexpr Burger burger;
constexpr Sandwitch sandwitch(true);
if (haveBacon(burger, sandwitch))
{return "nope";}
else{return "kosher";} }
```

Гарантии

Конструктор

Constexpr if

Consteval

Конструктор

```
struct Sandwitch{const bool bacon;
constexpr Sandwitch(bool haveBacon):bacon(haveBacon){};};


constexpr Sandwitch sandwitch(true);
```

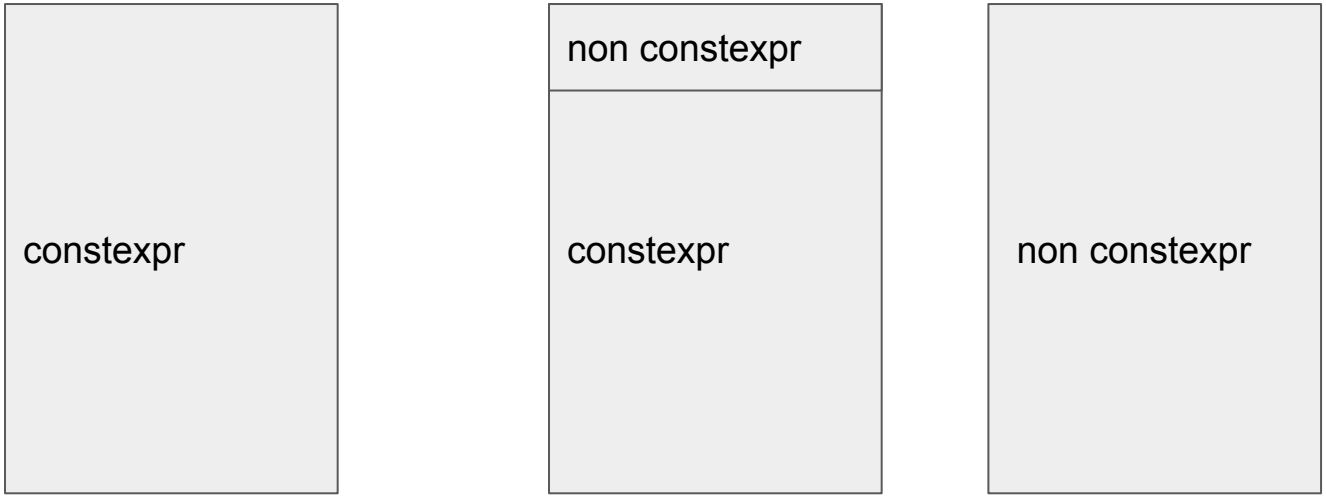
Constexpr if

```
if constexpr (haveBacon(burger, sandwitch))
{return ”nope”;}
else {return ”kosher”;}
```

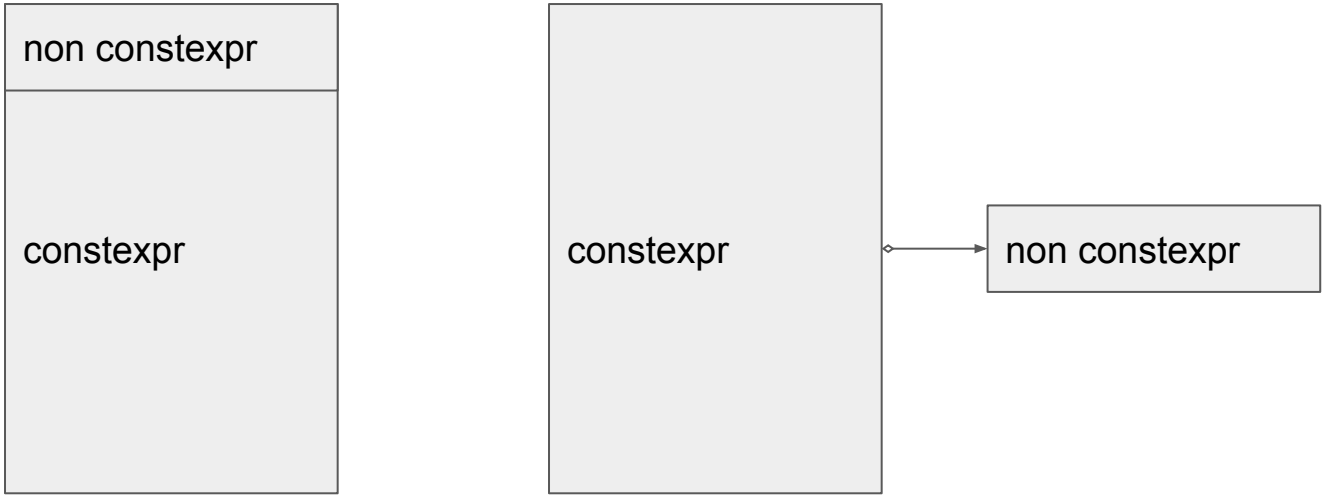
Consteval

```
template<typename ...Args>
consteval int haveBacon(Args&&... dish) {
   return (dish.bacon && ... );
}
```

Constexpr unconstexprable



Pimp constexpr



Пример

```
struct DynamicData

{
   DynamicData(int a,int b, int c):a_(a),b_(b),c_(c){}
   int a_,b_,c_;
};

struct StaticData
{
   constexpr StaticData(int d,int e,int f):d_(d),e_(e),f_(f){};
   int d_,e_,f_;
};
```

Пример

```
struct Logiclass
{
constexpr Logiclass (
    const StaticData& staticData,
    DynamicData* pDynamicData):
    staticData_(staticData),
    pDynamicData_(pDynamicData){};
StaticData staticData_;
DynamicData* pDynamicData_;

int aPlusd() const {return pDynamicData_->a_ + staticData_.d_ ;};
constexpr int ePlusd() const {return staticData_.e_ + staticData_.d_ ;};
};
```

Пример

```
DynamicData  dd{1,2,3};

int main(int a) {
dd.a_=a;
constexpr StaticData sd{4,5,6};
//fully constexpr class, even if it can operate with dynamic data
constexpr Logiclass lc(sd,&dd);
int one =  lc.aPlusd();
constexpr int two = lc.ePlusd();
return one + two;
}
```

Спасибо!

DINS®

Пример кода про который говорилось в конце

https://godbolt.org/z/_T_zAr