Code Generation

How the Lazy Becomes the Prolific

Nandakumar Edamana

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

Intro

Do We Program?

- We write code, but do we program?
- Theoretical CS is practical and IT is... boring?
- What problems do we solve in IT?
- Be lazy, and make them interesting!

Code Generators

This Counts

C in

```
return a + b;
```

asm out

```
movl    -4(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
...
ret
```

This Counts Too

High level? What about C Preprocessor?

```
#ifdef MORNING
  #define hello(x) Good morning, x!
#else
  #define hello(x) Hello, x!
#endif

hello(Geeks)
hello(Nerds)
```

This Counts Too

High level? What about C Preprocessor?

```
$ cpp hello.c |egrep -v '^#|^$'
Hello, Geeks!
Hello, Nerds!
```

```
$ cpp -DMORNING hello.c |egrep -v '^#|^$'
Good morning, Geeks!
Good morning, Nerds!
```

These Count Too

- Macros
- Generics
- `iota` in Go

Now We Are Talking

```yaml
openapi: 3.0.0
info:
  title: Calc API
  description: A simple API spec for OpenAPI demonstration.
  version: "1.0"

paths:
  /calc/sum:
    get:
      description: Returns the sum of the given numbers.
      parameters:
      - name: x
        in: query
        required: true
        schema:
          type: integer
      # "y" needs quoting because y is boolean yes in YAML
      - name: "y"
        in: query
        required: true
        schema:
          type: integer
      responses:
        '200':
          description: The sum.
  /calc/diff:
    get:
      description: Returns the difference of the given numbers.
```

Figure 1: OAPI yaml input, 43 lines

Now We Are Talking

```
57
58   // Parameter object where we will unmarshal all parameters from the context
59   var params GetCalcDiffParams
60
61   // ------------- Required query parameter "x" -------------
62   if paramValue := r.URL.Query().Get("x"); paramValue != "" {
63
64   } else {
65     siw.ErrorHandlerFunc(w, r, &RequiredParamError{ParamName: "x"})
66     return
67   }
68
69   err = runtime.BindQueryParameter("form", true, true, "x", r.URL.Query(), &params.X)
70   if err != nil {
71     siw.ErrorHandlerFunc(w, r, &InvalidParamFormatError{ParamName: "x", Err: err})
72     return
73   }
74
```

Figure 2: Go boilerplate, 351 lines

They Are Everywhere

From classic Unix tools to Go and k8s. We'll see.

Let's Begin

- No evangelism
- But they are there
- Possibilities? Limitations? Pitfalls?
- Specifics of any tool available in docs, sites, books, etc.
- What lacks is a bird's eye view, and we are doing it

Why (With Examples)

Passion

- I like declarative creation and generators
  - Graphviz, LaTeX, etc.
  - Custom text-to-HTML converter for my first book
  - Used annotated source and Doxygen to fill my project reports
- Makes us focus on the content and semantics
- The source is usually human readable text
  - Never goes obsolete
  - Easy to write tools for
  - Easy to search and update (automate with regex)
- Precision and control? Sometimes more, sometimes less.
- Reproducible (for graphics and infra; programming, even imperative, already is)

Lang vs Codegen

Why don't pick a supreme language?

- Languages like C, Go, Java and Python need to be general or domain-specific
  - Code generators can be problem-specific
  - Less tradeoffs
- Dilemma:
  - Great lang, lacks something (Go before generics)
  - Great lang, bad syntax
  - Bad lang, no choice (existing codebase)
- Who said code generators can't be languages?
  - m4, T4 and PHP
- Not just to overcome lang limitations (e.g.: OpenAPI)

Pros at a Glance

- Outsource boilerplating
- Get started easily and improve it later
- Keep the code consistent
- Memory safety -- example follows
- Security (OpenAPI validations)
- Shared code where librarification will be hard or less efficient
- Microservices (lots of shared code)

OpenAPI Workflow

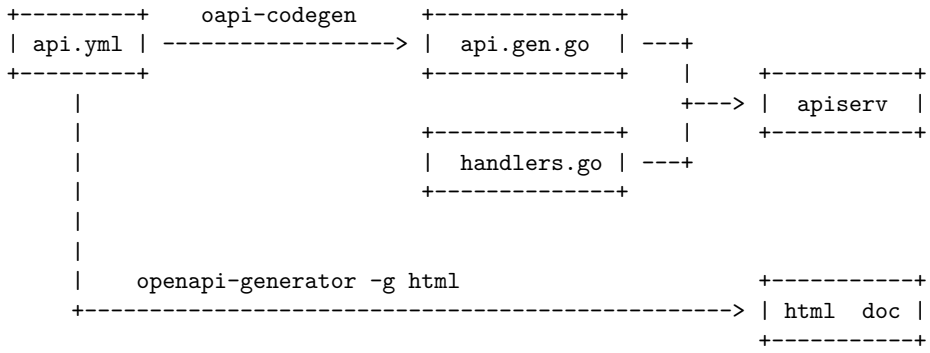

Figure 3: OpenAPI: spec to code and doc

Demo: Calc API -- Spec

Remember?

```yaml
openapi: 3.0.0
info:
  title: Calc API
  description: A simple API spec for OpenAPI demonstration.
  version: "1.0"

paths:
  /calc/sum:
    get:
      description: Returns the sum of the given numbers.
      parameters:
      - name: x
        in: query
        required: true
        schema:
          type: integer
      # "y" needs quoting because y is boolean yes in YAML
      - name: "y"
        in: query
        required: true
        schema:
          type: integer
      responses:
        '200':
          description: The sum.
  /calc/diff:
    get:
      description: Returns the difference of the given numbers.
```

Figure 4: OAPI yaml input, 43 lines

Demo: Calc API -- Handwritten

Hand-written handlers:

```
func (calc Calc) GetCalcSum(w http.ResponseWriter,
                            r *http.Request,
                            params GetCalcSumParams) {

  fmt.Fprintf(w, "%d\n", params.X + params.Y)
}

func (calc Calc) GetCalcDiff(w http.ResponseWriter,
                             r *http.Request,
                             params GetCalcDiffParams) {

  fmt.Fprintf(w, "%d\n", params.X - params.Y)
}
```

Demo: Calc API -- Auto-gen

Auto-generated structs for marshalling/unmarshalling:

```
// GetCalcDiffParams defines parameters for GetCalcDiff.
type GetCalcDiffParams struct {
  X int `form:"x" json:"x"`
  Y int `form:"y" json:"y"`
}

// GetCalcSumParams defines parameters for GetCalcSum.
type GetCalcSumParams struct {
  X int `form:"x" json:"x"`
  Y int `form:"y" json:"y"`
}
```

Demo: Calc API -- Auto-gen

Auto-generated interface:

```
// ServerInterface represents all server handlers.
type ServerInterface interface {

  // (GET /calc/diff)
  GetCalcDiff(w http.ResponseWriter,
              r *http.Request, params GetCalcDiffParams)

  // (GET /calc/sum)
  GetCalcSum(w http.ResponseWriter,
             r *http.Request, params GetCalcSumParams)
}
```

Demo: Calc API -- Auto-gen

Auto-generated validation and error messages in action:

```
$ curl 'localhost:8080/calc/diff?y=12'

parameter "x" in query has an error: \
value is required but missing

$ curl 'localhost:8080/calc/diff?x=10&y=someString'

parameter "y" in query has an error: \
value someString: an invalid integer: invalid syntax
```

OpenAPI Generator Types

- Server types, interfaces, validators, etc.
- Client SDK
- Documentation
- Database schema

Boilerplate: Even Inline

```
gtk_widget_set_tooltip_text(
  GTK_WIDGET(btn_run),
  "Run the command");

gtk_container_add(GTK_CONTAINER(vbox1),
  GTK_WIDGET(btn_run));

gtk_widget_show(GTK_WIDGET(btn_run));
gtk_widget_show_all(GTK_WIDGET(vbox1));
```

- Casting
- Error checks
- Initialization and finalization

Example: Stringify C Enum Manually

```c
// XXX Manually keep in sync with the array opstr
typedef enum Operation {
  OP_SUM,
  OP_DIFF,
  OP_LARGEST,
  OP_SMALLEST,
} Operation;

// XXX Manually keep in sync with the enum
const char * opstr[] = {
  "OP_SUM",
  "OP_DIFF",
  "OP_LARGEST",
  "OP_SMALLEST" };
```

Example: Stringify C Enum Manually

Problems?

- Chance for inconsistency is highly
- Incorrect debug messages and logs, without anybody realizing
- SEGFAULT if you forget to update `opstr` after growing the enum

Example: Stringify C Enum With sed

```
opstr.gen.h: main.c
  (echo '// Do not edit this file!' &&\
    echo 'const char * opstr[] = { ' &&\
    grep -E '^\s*OP_[A-Z]+,$$' main.c|\
      sed -E 's/^\s*(OP_[A-Z]+),$$/  "\1",/' &&\
    echo '};') > opstr.gen.h
```

So...

- Consistency -- guaranteed
- Better than handwritten code, if done right:
  - Maintainability
  - Security
- Productivity -- esp. if somebody else wrote the generator
- Fun, mayhem and fun again -- if you have to write the generator

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

In the Wild

Tools

- Compiler compilers: lex, yacc, etc.
- Transpilers
- Build system generators: Automake, CMake, qmake, etc.
- GUI builders: Glade, Qt Designer, etc.
- Configuration generators: `update-grub`
- From the Web world
  - CSS: SaSS, LESS, Stylus
  - k8s: kompose, kustomize, helm, etc.
- Interface/binding generators: SIP for Python (used for PyQt, wxPython, etc.)
  - PyBindGen, SWIG, etc.
- Flexible: Telosys from Eclipse (DB/model to any kind of code based on templates)
- `protoc` with gRPC Go plugin

Truth: I'm yet to try some of the above.

`go generate` and `stringer`

Remember the C enum stringification example?

Examples for `go generate` and `stringer`: go.dev/blog/generate

Other uses for `go generate`

> *... generating Unicode tables in the `unicode` package, creating efficient methods for encoding and decoding arrays in `encoding/gob`, producing time zone data in the `time` package, and so on.*

-- Rob Pike

m4

- A general-purpose macro processor fom 70s
- Not because the first, but because still alive
- History of macro processors:
  www.gnu.org/software/m4/manual/html_node/History.html
- m4 was used for *Raftor* preprocessor, a FORTRAN dialect

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

Considerations

Cons

- Ugly code - hard to read and debug
  - Extreme examples: output generated by lex and bison
  - Not even gdb can help
  - Ask: why ugly (efficiency?), and is it worth it?
- Less efficient
  - Tools have to be general; hard to optimize individual cases
- Bad error reporting (auto-generated lexers and parsers)
- You don't know what's happening behind the scenes
  - You still don't know what's happening behind the scenes when you write everything manually.

Some major compilers and interpreters use handwritten parsers.

Pro or Con?

- Declarative
  - Makes sure we have some kind of spec for our software
  - Same spec ruined to work aroud generator limitations

Legal

- What is the license of the generated code?
- GCC exception: www.gnu.org/licenses/gcc-exception-3.1.html
- GitHub Copilot controversy
  - Because it was trained using codebase under non-public-domain licenses
  - Not the kind of code generator we are talking about, BTW.

Never for Database

- Not something you want to put it in a pipeline

Tests Are Important

- What if a you forget to enable the OAPI validation flags or middleware?
- What if a flag changes in a future release of the generator?

That's why tests and assertions are important.

Auto-generate Tests?

- Could be a bad idea
- Go does this, but only to invoke our tests, AFAIK
- Auto-generated tests used to detect compiler bugs
  - Compare with the outputs of other compiler
  - Did I read it in a paper about CompCert (Coq-based C compiler)?
- If doing,
  - write a totally independent program
  - verify and lock the code manually

Making It Better

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

Tips

- Make it reproducible, if not too much trouble
- Prefer annotations over edits (will explain soon)
- Patch if needed
- Generated code: push or .gitignore?
- Linters and other static analysis tools
- `indent`, `gofmt`, etc.

kompose Experience

- What kompose is
- Convert once, forget the source -- okay.
- What if both docker-compose.yml and k8s are needed?
  - How to sync? Put in a pipeline.
  - What about the edits?

kompose Experience

What about the edits?

Maybe kompose annotations will help:

- `kompose.image-pull-policy`
- `kompose.image-pull-secret`, etc.
- etc.

kompose Experience

When annotations didn't help...

- k8s services for Docker Compose services
  - kompose generates them only if ports are exposed, AFAIK
  - Just wrote a custom script
- initContainers
  - Directly under `services` in docker-compose.yml, under `spec.template.spec.initContainers` in k8s deployment.yaml
  - Used patch files

```
40a41,45
>         "initContainers": [
>           {
>             "image": "REGISTRY/REPO/myapp",
>             "name": "myapp-con"
>           } ],
```

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

nguigen

What It Is

- A programming language
- A transpiler/code generator
- Self-hosted
- Targets:
  - C/GTK: usable
  - C++/Qt, Web: at infancy
  - Android: worked once, needs restart
  - Other: Go, Java, etc.
- Unreleased, but the plan is to go libre
- Similar (or not similar): Vala, Haxel, Nim, etc.

ngg for GUI: the Source

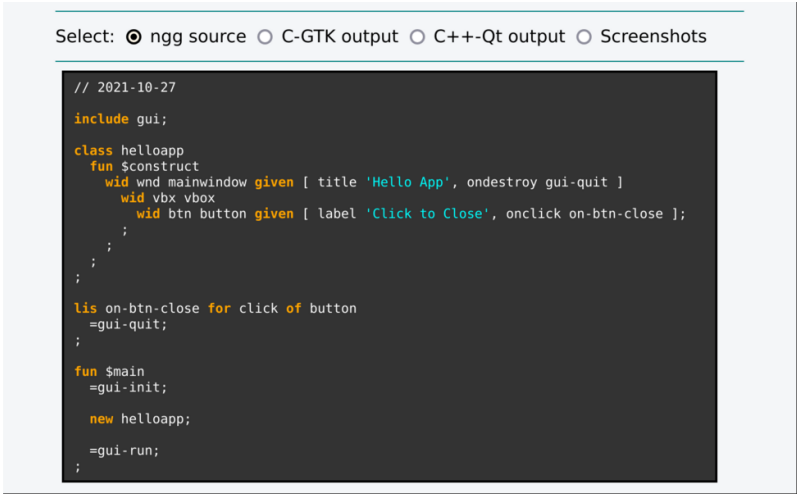

Figure 5: Source code of an ngg GUI program

ngg for GUI: C/GTK

Select: ○ ngg source ◉ C-GTK output ○ C++-Qt output ○ Screenshots

```
// Some stray code and boilerplate have been removed manually.
// The stray code will soon go away, although it never affects compilation or
// performance.

// hello.c:

void gui_quit()
{
    gtk_main_quit();
}

void helloapp_construct(helloapp * this )
{
    GtkWindow * wnd ;
    wnd = ((GtkWindow * ) gtk_window_new(GTK_WINDOW_TOPLEVEL));
    gtk_window_set_title(wnd, "Hello App");
    g_signal_connect(wnd, "destroy", G_CALLBACK(gui_quit), NULL);
    GtkBox * vbx ;
    vbx = gtk_box_new(GTK_ORIENTATION_VERTICAL, 0);
    gtk_container_add(GTK_CONTAINER((GtkWidget *) wnd), GTK_WIDGET(vbx));
    GtkButton * btn ;
    btn = ((GtkButton * ) gtk_button_new());
    gtk_container_add(GTK_CONTAINER((GtkWidget *) vbx), GTK_WIDGET(btn));
    gtk_button_set_label(btn, "Click to Close");
    g_signal_connect((GtkWidget *) btn, "clicked", G_CALLBACK(on_btn_close), this)
    gtk_widget_show(((GtkWidget * ) btn));
```

Figure 6: C/GTK generated code

ngg for GUI: C++/Qt

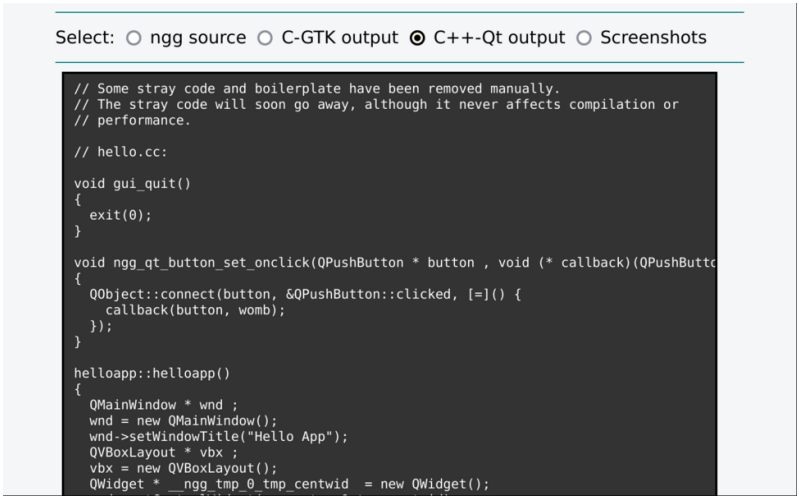

Figure 7: C++/Qt generated code

ngg for GUI: Screenshots

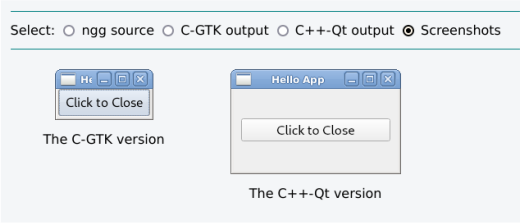

Figure 8: GTK and Qt screenshots

ngg for GUI: Auto-generated Test

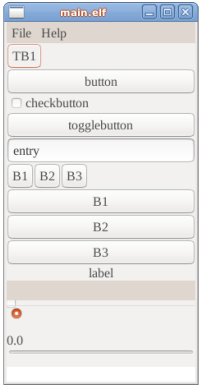

Figure 9: Auto-generated C/GTK test

ngg for GUI: Theeram

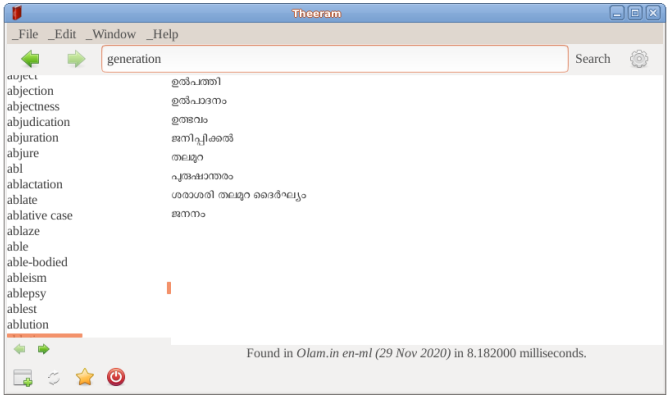

Figure 10: Theeram C to ngg migration, WIP

Origins of nguigen (1)

- Early dreams of own OS and programming languages
- Passion for C, but pitfalls and productivity issues

Origins of nguigen (2)

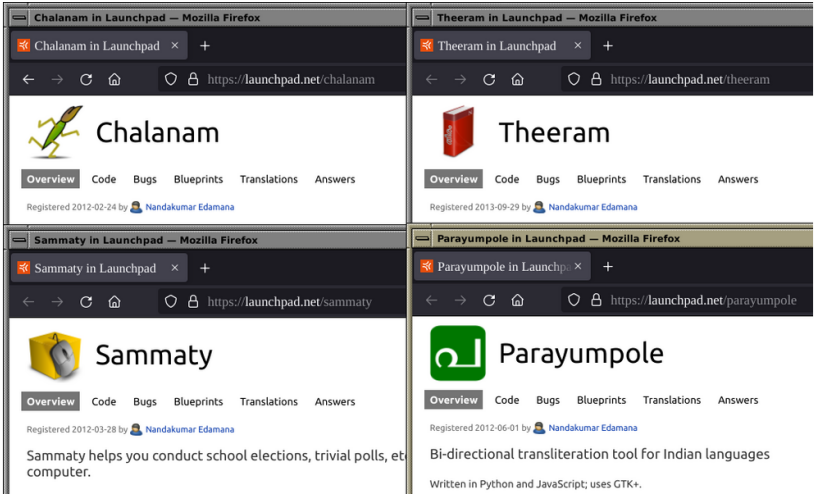

Figure 11: My Early PyGTK Works

Origins of nguigen (2)

- Publishing PyGTK apps since 2012 or before
- Thousands of school elections with Sammaty (20k downloads); not that I'm proud of
- Migration to C
- Parallel Web versions

Origins of nguigen (2)

- Desktop development frustrations
  - Differences in operating systems, desktops, different versions of the same GUI toolkit
  - Low productivity
  - How do I solve this without inventing one more framework?

Origins of nguigen (3)

C days...

What I ought to think about:

- Data structures
- Algorithms
- Programming methodologies and paradigms

What I was forced to think about:

- Overflows, underflows, memory leaks, etc.

What showed up always:

- SEGFAULT

nguigen: malloc() error check

ngg

```
local person new Person;
```

C output

```
Person * person = malloc(sizeof(Person));
if(person == NULL) {
  perror(NULL);
  exit(EXIT_FAILURE);
}
```

`local` offers:

Destruction and de-allocation of a local object Destruction and de-allocation of member object Setting invalid pointers back to NULL

Syntactic Sugar

Just an example...

ngg

```
if in targlang .[CXX, GO, JAVA]\Lang
  =puts/['Language supports generics.'];;
```

C output

```
if( targlang == LANG_CXX ||
    targlang == LANG_GO  ||
    targlang == LANG_JAVA )
{
  puts("Language supports generics.");
}
```

nguigen: Simplified Programming

- Compile-time semi-automatic memory management
  - Can I go the Rust way?
- No explicit pointers
- Casting to base class made easy in C (think: GTK)
- Restricted `for` loop
- Ergonomic syntax (less Shift key)
- Currently: class objs always in heap and struct objs always in stack

nguigen: Sophisticated Macro Processing

- Imlining -- inline before you get the C code
- Shadow classes -- wrap or customize external method calls
- Verbose lines
- Conditional code generation

Codegen in the nguigen Ecosystem

- nguigen is created with:
  - custom lexer generator
  - custom parser generator
  - AWK, sed, etc.: mapping-related code from tsv files (e.g.: data types)
  - Makefile parts generated with ngg itself and bash
- ngg to GTK, Qt, etc. mapping: interfaces and shadow classes with h2ngg and PHP

Data Type Mapping

bool	BOOL	_Bool	boolean	Boolean	bool
schar	SCHAR				
char	CHAR	char	-	-	byte
double	DOUBLE	double	double	-	float64
float	FLOAT	float			
int	INT	int	Integer	Int	
ldouble	LDOUBLE	long double			
long	LONG	long			

Figure 12: TSV file that maps ngg types to C, Java, etc. (WIP)

Data Type Mapping

```
const char * dtypes_c[48] = {"-", "-", " Bool", "", "char", "double", "float", "int", "long double", "long",
"void *", "void **", "short", "size_t", "ssize_t", "char *", "const char *", "unsigned char", "unsigned short",
"unsigned int", "unsigned long int", "unsigned long", "FILE *", "va_list", "const -", "const -", "const _Bool",
"const ", "const char", "const double", "const float", "const int", "const long double", "const long", "const
void *", "const void **", "const short", "const size_t", "const ssize_t", "const char *", "const const char *",
"const unsigned char", "const unsigned short", "const unsigned int", "const unsigned long int", "const unsigned
long", "const FILE *", "const va_list"};
const char * dtypes_go[48] = {"-", "-", "bool", "", "byte", "float64", "float", "int", "long double", "long",
"void *", "void **", "short", "int", "int", "char *", "string", "unsigned char", "unsigned short", "unsigned
int", "unsigned long int", "unsigned long", "FILE *", "va_list", "const -", "const -", "const bool", "const ",
"const byte", "const float64", "const float", "const int", "const long double", "const long", "const void *",
"const void **", "const short", "const int", "const int", "const char *", "const string", "const unsigned char",
"const unsigned short", "const unsigned int", "const unsigned long int", "const unsigned long", "const FILE *",
"const va_list"};
const char * dtypes_java[48] = {"", "", "boolean", "", "-", "double", "", "Integer", "", "", "", "", "", "int",
"int", "-", "String", "", "", "", "", "", "", "", "", "", "boolean", "", "-", "double", "", "Integer", "", "",
"", "", "", "int", "int", "-", "String", "", "", "", "", "", "", "", "};
const char * dtypes_kotlin[48] = {"", "", "Boolean", "", "-", "-", "", "Int", "", "", "", "", "", "-", "-", "",
"String", "", "", "", "", "", "", "", "", "", "Boolean", "", "-", "-", "", "Int", "", "", "", "", "", "-", "-",
"", "String", "", "", "", "", "", "", "", "};
```

Figure 13: Type strings generated from the TSV using AWK

Data Type Mapping

Type constants generated from the TSV using AWK:

```
typedef enum NggDtypetype {
  NGG_DTYPETYPE_UNKNOWN,
  NGG_DTYPETYPE_THIS_BAKE,
  NGG_DTYPETYPE_SUBTYPE,
  NGG_DTYPETYPE_BOOL,
  NGG_DTYPETYPE_SCHAR,
  NGG_DTYPETYPE_CHAR,
  NGG_DTYPETYPE_DOUBLE,
  NGG_DTYPETYPE_FLOAT,
  NGG_DTYPETYPE_INT,
  ...
```

Done So Far

- Parser generator (ngg + PHP)
- Lexer generator
  - gitlab.com/nandedamana/nlexgen
- Self-hosted transpiler
- h2ngg (we will see)
- Syntax highlighting in gedit

h2ngg

```
GtkWidget * gtk_font_selection_get_preview_entry (GtkFontSelection *fontsel);
            gtk_font_selection_get_family (GtkFontSelection *fontsel);
            gtk_font_selection_get_face (GtkFontSelection *fontsel);
gint gtk_font_selection_get_size (GtkFontSelection *fontsel);
gchar* gtk_font_selection_get_font_name (GtkFontSelection *fontsel);
gboolean gtk_font_selection_set_font_name (GtkFontSelection *fontsel,
const gchar* gtk_font_selection_get_preview_text (GtkFontSelection *fontsel);
void gtk_font_selection_set_preview_text (GtkFontSelection *fontsel,
GType gtk_font_selection_dialog_get_type (void) ;
GtkWidget *gtk_font_selection_dialog_new (const gchar *title);
GtkWidget *gtk_font_selection_dialog_get_ok_button (GtkFontSelectionDialog *fsd);
nandakumar@nandakumar-laptop:~/mnt/md127/nandakumar/my-works/software/nguigen/h2ngg/src/test-gtk-working$
```

```
extern fun gtk_font_selection_get_family gives PangoFontFamily takes fontsel GtkFontSelection;
extern fun gtk_font_selection_get_face gives PangoFontFace takes fontsel GtkFontSelection;
extern fun gtk_font_selection_get_size gives gint takes fontsel GtkFontSelection;
extern fun gtk_font_selection_get_font_name gives gchar takes fontsel GtkFontSelection;
extern fun gtk_font_selection_set_font_name gives gboolean takes fontsel GtkFontSelection, fontname gchar;
extern fun gtk_font_selection_get_preview_text gives gchar takes fontsel GtkFontSelection;
extern fun gtk_font_selection_set_preview_text takes fontsel GtkFontSelection, text gchar;
extern fun gtk_font_selection_dialog_get_type gives GType takes TODO-abstract-param;
extern fun gtk_font_selection_dialog_new gives GtkWidget takes title gchar;
extern fun gtk_font_selection_dialog_get_ok_button gives GtkWidget takes fsd GtkFontSelectionDialog;
extern fun gtk_font_selection_dialog_get_cancel_button gives GtkWidget takes fsd GtkFontSelectionDialog;
extern fun gtk_font_selection_dialog_get_font_selection gives GtkWidget takes fsd GtkFontSelectionDialog;
```

Figure 14: h2ngg converting GTK headers

Test Suit

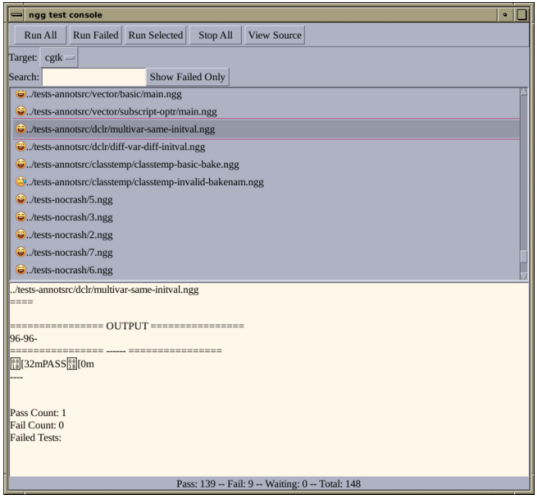

Figure 15: C/GTK Test Suit Runner written in nguigen (system GTK theme: *cdetheme-solaris*)

Intro
Code Generators
Why (With Examples)
In the Wild
Considerations
Making It Better
nguigen
The End

The End

Write Your Own

- There is no general solution; write your own!
- How?

Demo Repo

https://github.com/nandedamana/lazy-becomes-prolific

Currently contains `oapi-codegen-demo` and some trivial stuff included in this presentation.