C++'s Superpower

Matt Godbolt

MUREX

CODE RECKONS

Science to the CORE

C++'S SUPERPOWER

Image generated with Bing Image Creator powered by DALL·E

WHAT COULD IT BE?

- Ubiquity
- Performance
- Multi-paradigm
- Clear object lifetime

WHAT COULD IT BE?

- Ubiquity
- Performance
- Multi-paradigm
- Clear object lifetime

SURELY NOT?

- Undefined behaviour
- Wrong defaults
- Legacy support
- Confusing syntax

SURELY NOT?

- Undefined behaviour
- Wrong defaults
- Legacy support
- Confusing syntax

BACKWARDS COMPATIBILITY!

THE GOAL

- Take 25-year-old code
- Port to modern C++17

THE GOAL

- Take 25-year-old code
- Port to modern C++17

THE GOAL

- Take 25-year-old code
- Port to modern C++17

BUT MATT,
WHERE WILL YOU FIND 25-YEAR-OLD CODE?

Image generated with Bing Image Creator powered by DALL·E

1996

1996

1996

1996

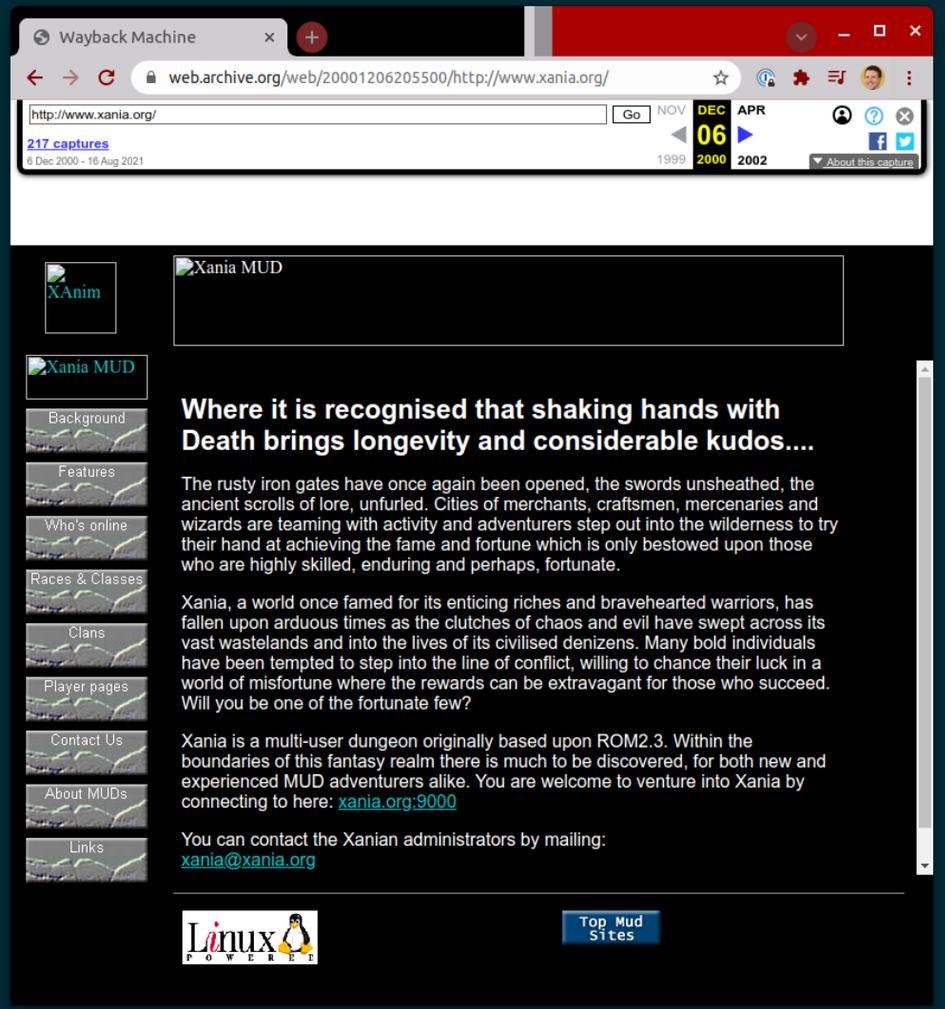
MUDS!

Image generated with Bing Image Creator powered by DALL·E

WHAT IS A MUD?

- Multi-User Dungeon
- Text-based
- Dungeons & Dragons
- Multiplayer
- Raw TCP clients

XANIA




```
The Temple Square

  This is the most southern end of the Temple of Isca. The surface of
the marble flooring has been chipped and worn by the thousands of feet
which have trampled through here since Midgaard was founded. To the
north, a fine set of curved steps opens before you, leading towards
the sacred altar. The centre of activity in this city is immediately
south of you, at Market Square. You hear faint sounds of battle high
above you and wonder why anyone would wish to fight in this fair city.

[Exits: north east south west up]
     A small white fountain gushes forth here.
A verminous gutter rat eyes you greedily and gnashes its teeth.
--> kill rat
Your stab does UNSPEAKABLE things to a gutter rat's head!
A gutter rat is DEAD!!
You receive 0 experience points.
A gutter rat hits the ground ... DEAD.
A gutter rat's severed head plops on the ground.
Etaine is unimpressed by your sacrifice but grants you a single gold coin.
Faramir gossips 'greetings!'
--> w
Entrance to Cleric's Guild

  The entrance hall is a small modest room, reflecting the true nature of
the Clerics.  The exit leads east to the temple square.  A small entrance to
the bar is in the northern wall.

[Exits: north east]
A vagabond is here, looking for victims.
A knight templar is guarding the entrance.
--> look vagabond
He looks pretty mean.
The vagabond is in excellent condition.

The vagabond is using:
<worn on body>      a hard leather jerkin
<worn around wrist> a leather bracer
<wielded>           a small sword

150 [||||||||||] 184/184 1609> w
150 [||||||||||] 184/184 1609> look vagabond
150 [||||||||||] 184/184 1609>
```

Image generated with [illegible]

WHAT IS A MUD?

Image generated with Bing Image Creator powered by DALL·E

WHAT IS A MUD?

Image generated with Bing Image Creator powered by DALL·E

GROUNDWORK

GROUNDWORK

Language	Files	Comments	Lines of Code
C	67	4524	50977
C++	6	54	699
Perl/make	8	180	717

GROUNDWORK

- `git`

GROUNDWORK

- `git`

GROUNDWORK

- `git`

GROUNDWORK

- git

GROUNDWORK

- git

GROUNDWORK

- git

GROUNDWORK

- git

DEMO!

THE C++-I-FICATION

```
$ rename .c .cpp *.c
$ rename .h .hpp *.h
$ ninja
...
```

POP QUIZ!

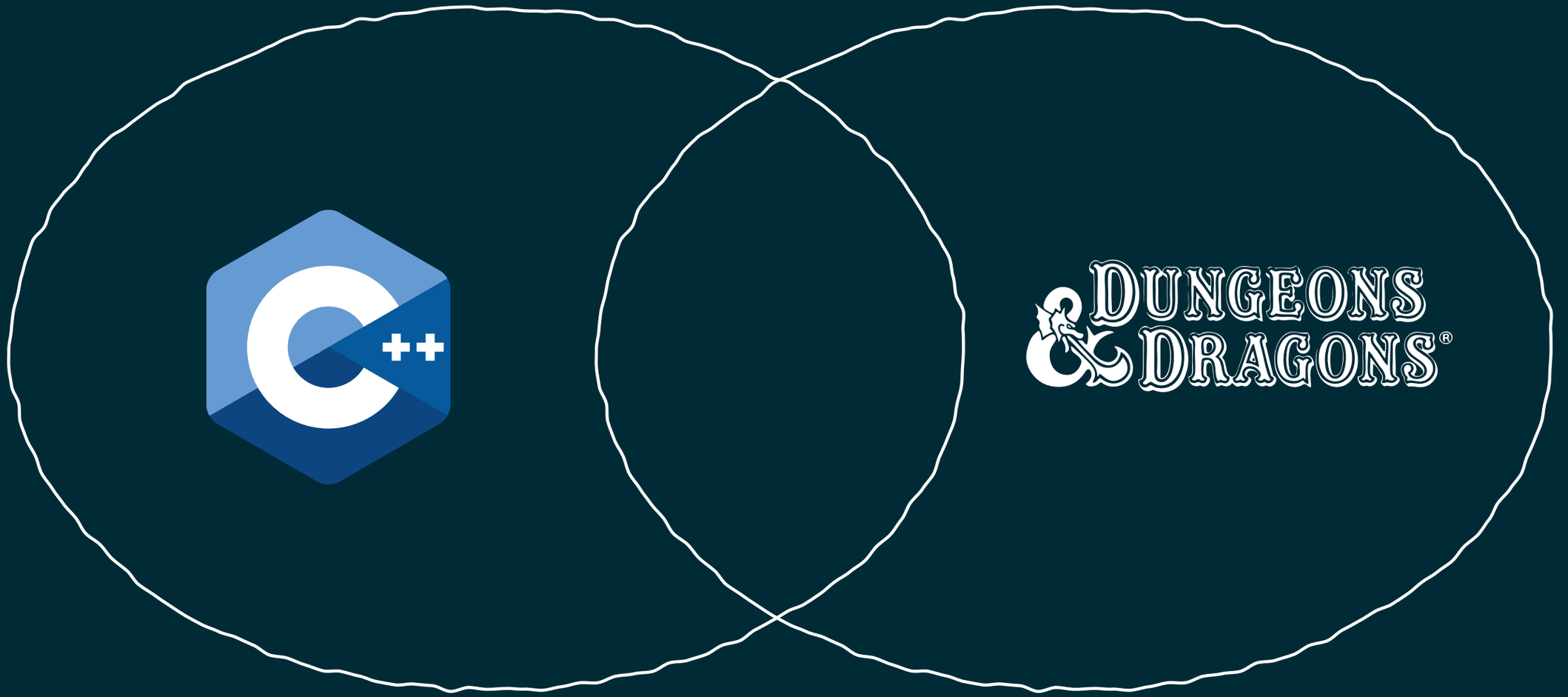


Image generated with [illegible]

POP QUIZ!

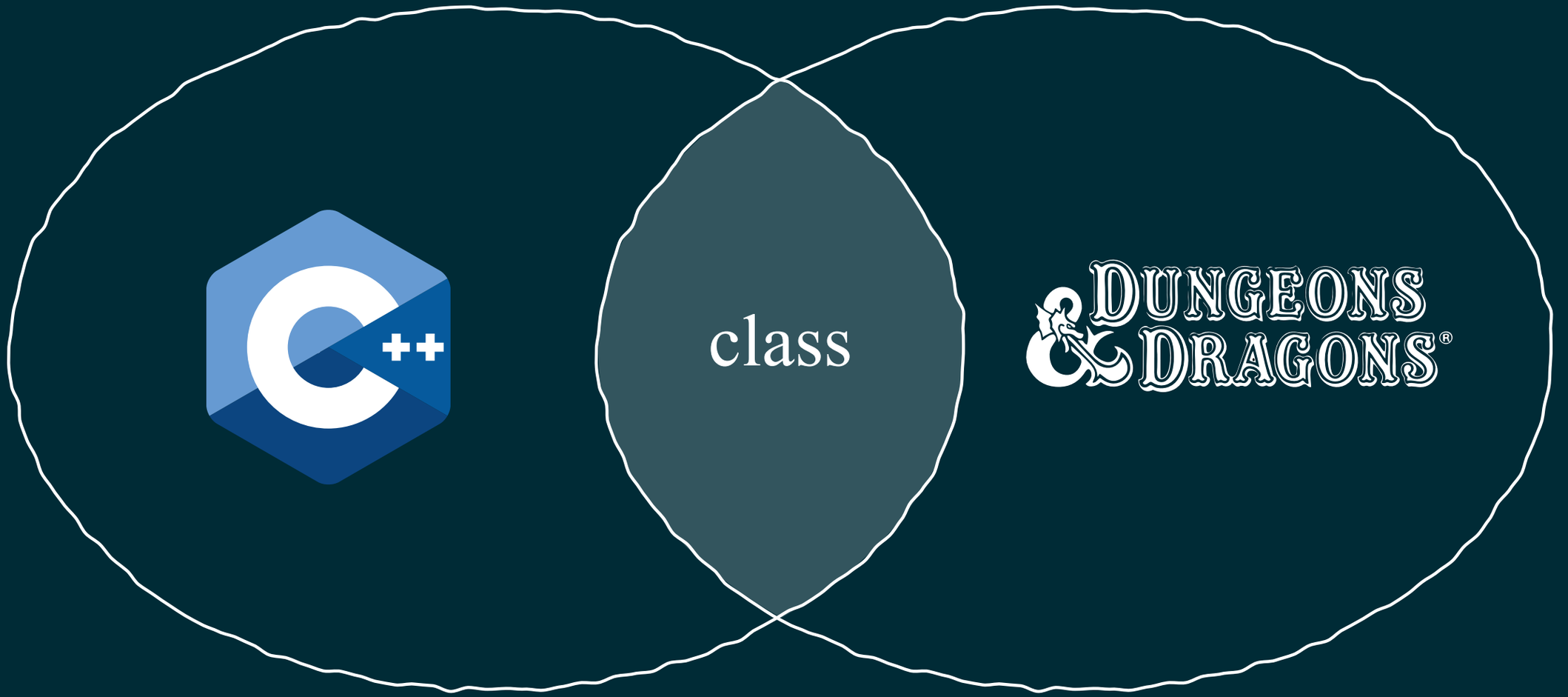


Image generated with [illegible]

POP QUIZ!

```
#define CLASS_MAGE 0
#define CLASS_CLERIC 1
// etc...
struct char_data {
  // ...
  int class;   /* what class this character is */
  // ...
};

// ...
/* Display the current flags */
void display_flags (char *template, CHAR_DATA *ch, int current_val);
```

PROCESS

- Small change
- Test
- Leave TODOs

TECHNIQUES

- CEDD
- JRIASWHDD
- TDD

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

HOW TO IMPROVE?

```
struct area_data {
  struct area_data *next;
  char *name;
  sh_int age;
  sh_int nplayer;
  bool empty;
  char *areaname;
  char *filename;
  int vnum;
};


typedef struct area_data AREA_DATA;
extern AREA_DATA *areas;
```

SIMPLE FIX

```
1 struct area_data {
2   struct area_data *next;
3   // ...
4 };
5 typedef struct area_data AREA_DATA;
```

SIMPLE FIX

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   // ...
4 };
```

CHANGING AN ELEMENT

- Build
- Run
- Manual test

CHANGING AN ELEMENT

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   char *name;
4   char *areaname;
5   char *filename;
6   // ...
7 };
```

CHANGING AN ELEMENT

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   std::string name;    // Look ma, I'm C++ now!
4   char *areaname;
5   char *filename;
6   // ...
7 };
```

CEDD

CEDD

CHANGING AN ELEMENT

```
 1 struct AREA_DATA {
 2   AREA_DATA *next;
 3   std::string name;
 4   // other members...
 5 };
 6
 7 void report_area(AREA_DATA *pArea) {
 8   char buf[MAX_STRING_LENGTH];
 9   sprintf(buf, "Area: %s\n", pArea->name);
10   // ...
11 }
```

CHANGING AN ELEMENT

```
 1 struct AREA_DATA {
 2   AREA_DATA *next;
 3   std::string name;
 4   // other members...
 5 };
 6
 7 void report_area(AREA_DATA *pArea) {
 8   char buf[MAX_STRING_LENGTH];
 9   sprintf(buf, "Area: %s\n", pArea->name);
10   // ...
11 }
```

CHANGING AN ELEMENT

```
 1 struct AREA_DATA {
 2   AREA_DATA *next;
 3   std::string name;
 4   // other members...
 5 };
 6
 7 void report_area(AREA_DATA *pArea) {
 8   char buf[MAX_STRING_LENGTH];
 9   sprintf(buf, "Area: %s\n", pArea->name);
10   // ...
11 }
```

CHANGING AN ELEMENT

```
 7 void report_area(AREA_DATA *pArea) {
 8   char buf[MAX_STRING_LENGTH];
 9   sprintf(buf, "Area: %s\n", pArea->name);
10   // ...
11 }
```

CHANGING AN ELEMENT

```
 7 void report_area(AREA_DATA *pArea) {
 8   char buf[MAX_STRING_LENGTH];
 9   sprintf(buf, "Area: %s\n", pArea->name.c_str());
10   // ...
11 }
```

CHANGING AN ELEMENT

```
1  /* Adds a line of text to the buffer, formatted. */
2 void buffer_addline_fmt(BUFFER *buffer, const char *text, ...);
```

CHANGING AN ELEMENT

```
1  /* Adds a line of text to the buffer, formatted. */
2 [[gnu::format(printf, 2, 3)]]
3 void buffer_addline_fmt(BUFFER *buffer, const char *text, ...);
```

CHANGING AN ELEMENT

JRIASWHDD

CHANGING AN ELEMENT

JRIASWHDD

CHANGING AN ELEMENT

```
==1090996==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 10 byte(s) in 1 object(s) allocated from:
 #0 in __interceptor_strdup (./xania/out/debug/bin/xania+0x1c3d017)
 #1 in parse_area(_IO_FILE*, char *) ./xania/src/db.cpp:349
 #2 in boot_db() ./xania/src/db.cpp:301
 ...
```

CHANGING AN ELEMENT

```
1  struct AREA_DATA {
2    AREA_DATA *next;
3    std::string name;
4    // ...
5  };
6
7  char *fread_string(FILE *fp); /* Read a string into a static buffer */
8  char *str_dup(char *str_to_duplicate); /* duplicates string with malloc */
9
10 void parse_area(AREA_DATA *pArea, FILE *fp) {
11   pArea->name = str_dup(fread_string(fp));
12   // ...rest of area parsing
13 }
```

CHANGING AN ELEMENT

```
1  struct AREA_DATA {
2    AREA_DATA *next;
3    std::string name;
4    // ...
5  };
6
7  char *fread_string(FILE *fp); /* Read a string into a static buffer */
8  char *str_dup(char *str_to_duplicate); /* duplicates string with malloc */
9
10 void parse_area(AREA_DATA *pArea, FILE *fp) {
11   pArea->name = str_dup(fread_string(fp));
12   // ...rest of area parsing
13 }
```

CHANGING AN ELEMENT

```
10 void parse_area(AREA_DATA *pArea, FILE *fp) {
11   pArea->name = str_dup(fread_string(fp));
12   // ...rest of area parsing
13 }
```

CHANGING AN ELEMENT

```
10 void parse_area(AREA_DATA *pArea, FILE *fp) {
11   pArea->name = fread_string(fp);
12   // ...rest of area parsing
13 }
```

AND SO ON

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   std::string name;
4   char *areaname;
5   char *filename;
6   // ...
7 };
```

AND SO ON

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   std::string name;
4   std::string areaname;
5   std::string filename;
6   // ...
7 };
```

WHAT'S NEXT?

```
1 struct AREA_DATA {
2   AREA_DATA *next;
3   std::string name;
4   std::string areaname;
5   std::string filename;
6   // ...
7 };
8
9 extern AREA_DATA *areas;
```

STL-I-FICATION

```
1 struct AREA_DATA {
2   std::string name;
3   std::string areaname;
4   std::string filename;
5   // ...
6 };
7
8 // TODO(#47): find a way to de-global this.
9 extern std::vector<AREA_DATA> areas;
```

STL-I-FICATION

```
1  AREA_DATA *current_area = NULL;
2
3  void load_area(FILE *fp) {
4    AREA_DATA area{};
5    parse_area(&area, fp);
6    areas.emplace_back(std::move(area));
7    current_area = &areas.back();
8  }
9
10 void load_room(FILE *fp) {
11   ROOM_DATA *room = /*...*/;
12   // ...
13   room->area = current_area;
14 }
```

STL-I-FICATION

```
1  AREA_DATA *current_area = NULL;
2
3  void load_area(FILE *fp) {
4    AREA_DATA area{};
5    parse_area(&area, fp);
6    areas.emplace_back(std::move(area));
7    current_area = &areas.back();
8  }
9
10 void load_room(FILE *fp) {
11   ROOM_DATA *room = /*...*/;
12   // ...
13   room->area = current_area;
14 }
```

STL-I-FICATION

```
1  AREA_DATA *current_area = NULL;
2
3  void load_area(FILE *fp) {
4    AREA_DATA area{};
5    parse_area(&area, fp);
6    areas.emplace_back(std::move(area));
7    current_area = &areas.back();
8  }
9
10 void load_room(FILE *fp) {
11   ROOM_DATA *room = /*...*/;
12   // ...
13   room->area = current_area;
14 }
```

STL-I-FICATION

```
ERROR: AddressSanitizer: heap-use-after-free on address
READ of size 2 at xxx188a2 thread T0
  #0 in reset_room(Room*) ./xania/src/db.cpp:1037
  #1 in reset_area() ./xania/src/db.cpp:985
...
xxx188a2 is located 34 bytes inside of 128-byte region [xxx18880,xxx18900)
freed by thread T0 here:
  #0 in operator delete(void*, unsigned long)
  #1 in __gnu_cxx::new_allocator<Area>::deallocate(Area*, unsigned long)
  #2 in std::allocator_traits<std::allocator<Area> >::deallocate...
...
previously allocated by thread T0 here:
  #0 in operator new(unsigned long)
  #1 in __gnu_cxx::new_allocator<Area>::allocate(unsigned long, void const*)
  #2 in std::allocator_traits<std::allocator<Area> >::allocate...
...
SUMMARY: heap-use-after-free ./xania/src/db.cpp:1037 in reset_room(Room*)
```

STL-I-FICATION

```
1  struct ROOM_DATA {
2    //...
3    AREA_DATA *area;
4    //...
5  };
6
7  void reset_room(ROOM_DATA *room) {
8    if (room->area->nplayer > 0) {
9      return;
10   }
11   // ...
12 }
```

STL-I-FICATION

```
1  struct ROOM_DATA {
2    //...
3    AREA_DATA *area;
4    //...
5  };
6
7  void reset_room(ROOM_DATA *room) {
8    if (room->area->nplayer > 0) {
9      return;
10   }
11   // ...
12 }
```

STL-I-FICATION

```
1  struct ROOM_DATA {
2    //...
3    AREA_DATA *area;
4    //...
5  };
6
7  void reset_room(ROOM_DATA *room) {
8    if (room->area->nplayer > 0) {
9      return;
10   }
11   // ...
12 }
```

STL-I-FICATION

```
1  struct ROOM_DATA {
2    //...
3    AREA_DATA *area;
4    //...
5  };
6
7  void reset_room(ROOM_DATA *room) {
8    if (room->area->nplayer > 0) {
9      return;
10   }
11   // ...
12 }
```

STL-I-FICATION

```
 1 struct AREA_DATA {
 2   std::string name;
 3   std::string areaname;
 4   std::string filename;
 5   // ...
 6 };
 7
 8 // TODO(#47): find a way to de-global this.
 9 // TODO(#48): try and remove need for pointer stability.
10 std::vector<std::unique_ptr<AREA_DATA>> areas;
```

STL-I-FICATION

```
 1 struct AREA_DATA {
 2   std::string name;
 3   std::string areaname;
 4   std::string filename;
 5   // ...
 6 };
 7
 8 // TODO(#47): find a way to de-global this.
 9 // TODO(#48): try and remove need for pointer stability.
10 std::vector<std::unique_ptr<AREA_DATA>> areas;
```

CLASS-I-FY

- `struct` → `class`
- Make all members `public:`
- One at a time:
  - make private
  - add accessors

CLASS-I-FY

- `struct` → `class`
- Make all members `public:`
- One at a time:
  - make private
  - add accessors

CLASS-I-FY

```
1 struct AREA_DATA {
2   std::string name;
3   std::string areaname;
4   std::string filename;
5   // ...
6 };
```

CLASS-I-FY

```
1 class AREA_DATA {
2 public:
3   std::string name;
4   std::string areaname;
5   std::string filename;
6   // ...
7 };
```

CLASS-I-FY

```
 1 class AREA_DATA {
 2   std::string name_;
 3
 4 public:
 5   std::string areaname;
 6   std::string filename;
 7
 8   [[nodiscard]] const std::string &name() const { return name_; }
 9   // ...
10 };
```

CLASS-I-FY

```
1  class AREA_DATA {
2    std::string name_;
3
4  public:
5    std::string areaname;
6    std::string filename;
7
8    [[nodiscard]] const std::string &name() const { return name_; }
9
10   void parse(FILE *fp);
11   // ...
12 };
```

CLASS-I-FY

```
1  class AREA_DATA {
2    std::string name_;
3    std::string area_name_;
4
5  public:
6    std::string filename;
7
8    [[nodiscard]] const std::string &name() const { return name_; }
9    [[nodiscard]] const std::string &areaname() const { return area_name_; }
10
11   void parse(FILE *fp);
12   // ...
13 };
```

CLASS-I-FY

```
1  class AREA_DATA {
2    std::string name_;
3    std::string area_name_;
4    std::string filename_;
5
6  public:
7
8    [[nodiscard]] const std::string &name() const { return name_; }
9    [[nodiscard]] const std::string &areaname() const { return area_name_; }
10   [[nodiscard]] const std::string &filename() const { return filename_; }
11
12   void parse(FILE *fp);
13   // ...
14 };
```

CLASS-I-FY

- Move functionality
- `parse_area` → `Area::parse`
- Write tests!
- Rename fields & accessors

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

CLASS-I-FIED

```
 1 class Area {
 2   std::string description_;
 3   int num_players_{};
 4   bool empty_since_last_reset_{};
 5   int min_level_{0};
 6   int max_level_{MAX_LEVEL};
 7   std::string short_name_;
 8
 9   Area() = default;
10   void reset();
11
12 public:
13   // ...continued on next panel -->
```

```
14   static Area parse(
15     FILE *fp,
16     std::string filename);
17
18   void player_entered();
19   void player_left();
20   void update();
21
22   [[nodiscard]]
23   const auto &short_name() const {
24     return short_name_;
25   }
26   // etc...
27 };
```

TEST

```
1  TEST_CASE("area loading") {
2    test::MemFile fp(R"(ignored~
3  Short name~
4  { 1 50} TheMoog Some kind of area~
5  6200 6399)");
6
7    SECTION("should parse") {
8      auto area = Area::parse(fp.file(), "bob");
9      CHECK(area.filename() == "bob");
10     CHECK(area.short_name() == "Short name");
11     CHECK(area.description() == "{ 1 50} TheMoog Some kind of area");
12     CHECK(area.min_level() == 1);
13     CHECK(area.max_level() == 50);
14   }
15 }
```

TEST

```
1  TEST_CASE("area loading") {
2    test::MemFile fp(R"(ignored~
3  Short name~
4  { 1 50} TheMoog Some kind of area~
5  6200 6399)");
6
7    SECTION("should parse") {
8      auto area = Area::parse(fp.file(), "bob");
9      CHECK(area.filename() == "bob");
10     CHECK(area.short_name() == "Short name");
11     CHECK(area.description() == "{ 1 50} TheMoog Some kind of area");
12     CHECK(area.min_level() == 1);
13     CHECK(area.max_level() == 50);
14   }
15 }
```

TEST

```
1  TEST_CASE("area loading") {
2    test::MemFile fp(R"(ignored~
3  Short name~
4  { 1 50} TheMoog Some kind of area~
5  6200 6399)");
6
7    SECTION("should parse") {
8      auto area = Area::parse(fp.file(), "bob");
9      CHECK(area.filename() == "bob");
10     CHECK(area.short_name() == "Short name");
11     CHECK(area.description() == "{ 1 50} TheMoog Some kind of area");
12     CHECK(area.min_level() == 1);
13     CHECK(area.max_level() == 50);
14   }
15 }
```

TEST

```
1  TEST_CASE("area loading") {
2    test::MemFile fp(R"(ignored~
3  Short name~
4  { 1 50} TheMoog Some kind of area~
5  6200 6399)");
6
7    SECTION("should parse") {
8      auto area = Area::parse(fp.file(), "bob");
9      CHECK(area.filename() == "bob");
10     CHECK(area.short_name() == "Short name");
11     CHECK(area.description() == "{ 1 50} TheMoog Some kind of area");
12     CHECK(area.min_level() == 1);
13     CHECK(area.max_level() == 50);
14   }
15 }
```

STRING FORMATTING

STRING FORMATTING

```
1 char buf[MAX_STRING_LENGTH];
2
3 sprintf(
4   buf,
5   "Your gain is: %d/%d hp, %d/%d m, %d/%d mv %d/%d prac.\n",
6   add_hp, ch->max_hit, add_mana, ch->max_mana,
7   add_move, ch->max_move, add_prac, ch->practice);
8 send_to_player(ch, buf);
```

STRING FORMATTING

```
1 char buf[MAX_STRING_LENGTH];
2
3 snprintf(
4   buf,
5   sizeof(buf),
6   "Your gain is: %d/%d hp, %d/%d m, %d/%d mv %d/%d prac.\n",
7   add_hp, ch->max_hit, add_mana, ch->max_mana,
8   add_move, ch->max_move, add_prac, ch->practice);
9 send_to_player(ch, buf);
```

STRING FORMATTING

```
1 void Char::send_to(const char *fmt, ...);
2
3 ch->send_to(
4   "Your gain is: %d/%d hp, %d/%d m, %d/%d mv %d/%d prac.\n",
5   add_hp, ch->max_hit, add_mana, ch->max_mana,
6   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

```
1 [[gnu::format(printf, 1, 2)]]
2 void Char::send_to(const char *fmt, ...);
3
4 ch->send_to(
5   "Your gain is: %d/%d hp, %d/%d m, %d/%d mv %d/%d prac.\n",
6   add_hp, ch->max_hit, add_mana, ch->max_mana,
7   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

```
1  void Char::send_to(std::string_view txt);
2  template <typename... Args>
3  void Char::send_to(std::string_view format, Args &&... args) const {
4    return send_to(fmt::format(format, std::forward<Args>(args)...));
5  }
6
7  ch->send_to(
8    "Your gain is: {}/{} hp, {}/{} m, {}/{} mv {}/{} prac.",
9    add_hp, ch->max_hit, add_mana, ch->max_mana,
10   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

```
1  void Char::send_to(std::string_view txt);
2  template <typename... Args>
3  void Char::send_to(std::string_view format, Args &&... args) const {
4    return send_to(fmt::format(format, std::forward<Args>(args)...));
5  }
6
7  ch->send_to(
8    "Your gain is: {}/{} hp, {}/{} m, {}/{} mv {}/{} prac.",
9    add_hp, ch->max_hit, add_mana, ch->max_mana,
10   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

```
1  void Char::send_to(std::string_view txt);
2  template <typename... Args>
3  void Char::send_to(std::string_view format, Args &&... args) const {
4    return send_to(fmt::format(format, std::forward<Args>(args)...));
5  }
6
7  ch->send_to(
8    "Your gain is: {}/{} hp, {}/{} m, {}/{} mv {}/{} prac.",
9    add_hp, ch->max_hit, add_mana, ch->max_mana,
10   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

```
1  void Char::send_to(std::string_view txt);
2  template <typename... Args>
3  void Char::send_to(std::string_view format, Args &&... args) const {
4    return send_to(fmt::format(format, std::forward<Args>(args)...));
5  }
6
7  ch->send_to(
8    "Your gain is: {}/{} hp, {}/{} m, {}/{} mv {}/{} prac.",
9    add_hp, ch->max_hit, add_mana, ch->max_mana,
10   add_move, ch->max_move, add_prac, ch->practice);
```

STRING FORMATTING

STRING FORMATTING

STRING FORMATTING

STRING FORMATTING

STRING FORMATTING

STRING MANIPULATION

STRING MANIPULATION

- Capitalize first letter
- To uppercase/lowercase
- Whitespace trimming
- Case insensitive matches
- Prefixes/suffixes
- Parsing/tokenising

STRING MANIPULATION

```c
char *capitalize(const char *str) {
  static char strcap[MAX_STRING_LENGTH];
  int i;

  for (i = 0; str[i] != '\0'; i++)
    strcap[i] = LOWER(str[i]);
  strcap[i] = '\0';
  strcap[0] = UPPER(strcap[0]);
  return strcap;
}
```

STRING MANIPULATION

```c
char *capitalize(const char *str) {
  static char strcap[MAX_STRING_LENGTH];
  int i;

  for (i = 0; str[i] != '\0'; i++)
    strcap[i] = LOWER(str[i]);
  strcap[i] = '\0';
  strcap[0] = UPPER(strcap[0]);
  return strcap;
}
```

STRING MANIPULATION

```c
char *capitalize(const char *str) {
  static char strcap[MAX_STRING_LENGTH];
  int i;

  for (i = 0; str[i] != '\0'; i++)
    strcap[i] = LOWER(str[i]);
  strcap[i] = '\0';
  strcap[0] = UPPER(strcap[0]);
  return strcap;
}
```

STRING MANIPULATION

```c
char *capitalize(const char *str) {
  static char strcap[MAX_STRING_LENGTH];
  int i;

  for (i = 0; str[i] != '\0'; i++)
    strcap[i] = LOWER(str[i]);
  strcap[i] = '\0';
  strcap[0] = UPPER(strcap[0]);
  return strcap;
}
```

STRING MANIPULATION

```
1 SECTION("capitalize") {
2   CHECK(capitalize("") == ""s);
3   CHECK(capitalize("a") == "A"s);
4   CHECK(capitalize("A") == "A"s);
5   CHECK(capitalize("a monkey") == "A monkey"s);
6   CHECK(capitalize("A MONKEY") == "A monkey"s);
7   CHECK(capitalize("a MonkeY") == "A monkey"s);
8 }
```

STRING MANIPULATION

```
1 SECTION("capitalize") {
2   CHECK(capitalize("") == ""s);
3   CHECK(capitalize("a") == "A"s);
4   CHECK(capitalize("A") == "A"s);
5   CHECK(capitalize("a monkey") == "A monkey"s);
6   CHECK(capitalize("A MONKEY") == "A monkey"s);
7   CHECK(capitalize("a MonkeY") == "A monkey"s);
8 }
```

STRING MANIPULATION

```
1 SECTION("capitalize") {
2   CHECK(capitalize("") == ""s);                    // Zero, Boundaries
3   CHECK(capitalize("a") == "A"s);                  // One
4   CHECK(capitalize("A") == "A"s);
5   CHECK(capitalize("a monkey") == "A monkey"s);   // Many
6   CHECK(capitalize("A MONKEY") == "A monkey"s);
7   CHECK(capitalize("a MonkeY") == "A monkey"s);
8 }
```

ZOMBIES

Image generated with Bing Image Creator

STRING MANIPULATION

```
1 std::string capitalize(const char *str) {
2   std::string result;
3   result.resize(strlen(str));
4   for (int i = 0; i < strlen(str); i++)
5     result[i] = std::tolower(str[i]);
6   result[0] = std::toupper(result[0]);
7   return result;
8 }
```

STRING MANIPULATION

```
1 std::string capitalize(const char *str) {
2   std::string result;
3   result.resize(strlen(str));
4   for (int i = 0; i < strlen(str); i++)
5     result[i] = std::tolower(str[i]);
6   result[0] = std::toupper(result[0]);
7   return result;
8 }
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

```cpp
std::string capitalize(const char *str) {
  if (!*str)
    return "";

  std::string result;
  std::transform(
    str, str + strlen(str),
    std::back_inserter(result),
    [](char c) -> char {
      return std::tolower(c);
    }
  );
  result[0] = std::toupper(result[0]);
  return result;
}
```

STRING MANIPULATION

ENTER `string_view` AND RANGES V3

STRING MANIPULATION

ENTER `string_view` AND RANGES V3

STRING MANIPULATION

RANGES

`range ≡ [begin, end)`

STRING MANIPULATION

RANGES

`range ≡ [begin, end)`

STRING MANIPULATION

RANGES

```
std::some_algo(begin(container), end(container));
```

→

```
ranges::some_algo(container);
```

STRING MANIPULATION

RANGES

- Lazy evaluation with `ranges::view` (`transform`, `concat`, etc)
- Composable with `operator |`

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
        [](const auto &pair) -> char {
          return pair.first == 0 ? std::toupper(pair.second)
                                 : std::tolower(pair.second);
        }
      )
    | ranges::to<std::string>;
}
```

STRING MANIPULATION

```cpp
auto capitalize(std::string_view text) {
  return text
    | ranges::view::enumerate
    | ranges::view::transform(
      [](const auto &pair) -> char {
        return pair.first == 0 ? std::toupper(pair.second)
                               : std::tolower(pair.second);
      }
    )
    | ranges::to<std::string>;
}
```

```
std::string_view("hi!") →
range<tuple<int, char>>((0, 'h'), (1, 'i'), (2, '!')) →
range<char>('H', 'i', '!') →
std::string("Hi!")
```

STRING MANIPULATION

```
1 auto capitalize(std::string_view text) {
2   return ranges::view::concat(
3     text | ranges::view::take(1) | ranges::view::transform(toupper),
4     text | ranges::view::drop(1) | ranges::view::transform(tolower)
5   ) | ranges::to<std::string>;
6 }
```

STRING MANIPULATION

```
1 auto capitalize(std::string_view text) {
2   return ranges::view::concat(
3     text | ranges::view::take(1) | ranges::view::transform(toupper),
4     text | ranges::view::drop(1) | ranges::view::transform(tolower)
5   ) | ranges::to<std::string>;
6 }
```

STRING MANIPULATION

```
1 auto capitalize(std::string_view text) {
2   return ranges::view::concat(
3     text | ranges::view::take(1) | ranges::view::transform(toupper),
4     text | ranges::view::drop(1) | ranges::view::transform(tolower)
5   ) | ranges::to<std::string>;
6 }
```

STRING MANIPULATION

```
1 auto capitalize(std::string_view text) {
2   return ranges::view::concat(
3     text | ranges::view::take(1) | ranges::view::transform(toupper),
4     text | ranges::view::drop(1) | ranges::view::transform(tolower)
5   ) | ranges::to<std::string>;
6 }
```

STRING MANIPULATION

```
1 auto capitalize(std::string_view text) {
2   return ranges::view::concat(
3     text | ranges::view::take(1) | ranges::view::transform(toupper),
4     text | ranges::view::drop(1) | ranges::view::transform(tolower)
5   ) | ranges::to<std::string>;
6 }
```

STRING MANIPULATION

CASE-INSENSITIVE `strstr`

STRING MANIPULATION

```
bool has_prefix(const char *astr, const char *bstr) { /* ... */ }

bool matches_inside(const char *astr, const char *bstr) {
    char c0;
    if ((c0 = LOWER(astr[0])) == '\0')
        return true;

    int sstr1 = strlen(astr);
    int sstr2 = strlen(bstr);

    for (int ichar = 0; ichar <= sstr2 - sstr1; ichar++) {
        if (c0 == LOWER(bstr[ichar]) && has_prefix(astr, bstr + ichar))
            return true;
    }
    return false;
}
```

STRING MANIPULATION

```cpp
bool matches_inside(std::string_view needle, std::string_view haystack) {
  auto needle_low = needle | ranges::views::transform(tolower);
  auto haystack_low = haystack | ranges::views::transform(tolower);
  return !ranges::search(haystack_low, needle_low).empty();
}
```

STRING MANIPULATION

```
1 bool matches_inside(std::string_view needle, std::string_view haystack) {
2   auto needle_low = needle | ranges::views::transform(tolower);
3   auto haystack_low = haystack | ranges::views::transform(tolower);
4   return !ranges::search(haystack_low, needle_low).empty();
5 }
```

STRING MANIPULATION

```cpp
bool matches_inside(std::string_view needle, std::string_view haystack) {
  auto needle_low = needle | ranges::views::transform(tolower);
  auto haystack_low = haystack | ranges::views::transform(tolower);
  return !ranges::search(haystack_low, needle_low).empty();
}
```

STRING MANIPULATION

```cpp
bool matches_inside(std::string_view needle, std::string_view haystack) {
  auto needle_low = needle | ranges::views::transform(tolower);
  auto haystack_low = haystack | ranges::views::transform(tolower);
  return !ranges::search(haystack_low, needle_low).empty();
}
```

MORE FUN WITH RANGES

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

STRING FORMATTING & RANGES

```
ch->send_line(
  "You can train: {}.",
  fmt::join(

    ranges::views::concat(
      all_stats | ranges::views::filter([&](auto stat) {
        return ch->perm_stat[stat] < get_max_train(ch, stat);
      }) | ranges::views::transform(to_short_string),

      always_trainable   // "hp" and "mana"
    ),

    " "sv
  )
);
```

```
You can train: str dex con hp mana.
```

`std::array` AND RANGES

```cpp
int armor[MAX_AC];
// ...
for (int i = 0; i < MAX_AC; ++i)
  armor[i] = 0;
// ...
for (int i = 0; i < MAX_AC; ++i)
  armor[i] = -1;
```

```cpp
std::array<int, MAX_AC> armour {};
// ...
ranges::fill(armour, -1);
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

RANGES

```
1  void Char::yell(std::string_view exclamation) const {
2    send_line("You yell '{}'", exclamation);
3
4    ranges::for_each(
5      descriptors().all_but(*this)
6      | DescriptorFilter::same_area(*this)
7      | DescriptorFilter::to_character(),
8      [&](auto &victim) {
9        victim.send_line("{} yells '{}'", describe_for(victim), exclamation);
10     }
11   );
12 }
```

BUT WAIT THERE'S MORE...

Image generated with [illegible]

NOT COVERED HERE

- Strong types & `enums`
- Bit field enums
- `std::function`
- `std::optional`
- `std::variant`

REGRETS

- Fuzzing `libfuzz`
- TODO: unglobalisation
- C++20

CONCLUSION

CONCLUSION

Language	Files	Comments	Lines of Code
C++	151	3839	42117 (down from 50k)
CMake	10	0	93

WHAT HELPED

- Compiler warnings
- Sanitisers
- CEDD & TDD (ish)

WHAT HELPED

- Compiler warnings
- Sanitisers
- CEDD & TDD (ish)

BENEFITS OF MODERNISING YOUR CODE

- Easier to find and fix bugs
- More understandable code
- Keeps coders happy

BENEFITS OF MODERNISING YOUR CODE

- Easier to find and fix bugs
- More understandable code
- Keeps coders happy

CONCLUSION

THE PAST

CONCLUSION

THE NOW

CONCLUSION

THE FUTURE

THANKS TO

- Rob Snell, Phil Norman & the other Xania developers
- The DIKU team: Katja Nyboe, Tom Madsen, Hans Henrik Staerfeldt, Michael Seifert & Sebastian Hammer
- The ROM developer Russ Taylor
- Hana Dusíková
- Aquatic
- C++ Denver

`~Talk() {`

- Source on Github: mattgodbolt/xania
- **Beta** at `telnet mud.xania.org 9000`
- Maybe your code will still compile in 2046!

`}`

~Talk() {

THANK YOU!

HAPPY 40TH BIRTHDAY BBC MICRO

virtual.bbcmic.ro

}

OTHER TRICKS

PARSING

```
1  typedef void DO_FUN(Char *ch, const char *argument);
2
3  void init_commands() { /* populates some kind of map */ }
4  DO_FUN *find_command(const char *) { /* looks up in map */ }
5
6  void parse(Char *ch, const char *line) {
7    char command[MAX_LINE_LENGTH];
8    char *remainder = one_argument(line, command);
9    DO_FUN *func = find_command(command);
10   if (!func) {
11     send_to_char(ch, "Huh?\n");
12     return;
13   }
14   (*func)(ch, remainder);
15 }
```

PARSING

```
1  typedef void DO_FUN(Char *ch, const char *argument);
2
3  void init_commands() { /* populates some kind of map */ }
4  DO_FUN *find_command(const char *) { /* looks up in map */ }
5
6  void parse(Char *ch, const char *line) {
7    char command[MAX_LINE_LENGTH];
8    char *remainder = one_argument(line, command);
9    DO_FUN *func = find_command(command);
10   if (!func) {
11     send_to_char(ch, "Huh?\n");
12     return;
13   }
14   (*func)(ch, remainder);
15 }
```

PARSING

```
1  typedef void DO_FUN(Char *ch, const char *argument);
2
3  void init_commands() { /* populates some kind of map */ }
4  DO_FUN *find_command(const char *) { /* looks up in map */ }
5
6  void parse(Char *ch, const char *line) {
7    char command[MAX_LINE_LENGTH];
8    char *remainder = one_argument(line, command);
9    DO_FUN *func = find_command(command);
10   if (!func) {
11     send_to_char(ch, "Huh?\n");
12     return;
13   }
14   (*func)(ch, remainder);
15 }
```

PARSING

```
1  typedef void DO_FUN(Char *ch, const char *argument);
2
3  void init_commands() { /* populates some kind of map */ }
4  DO_FUN *find_command(const char *) { /* looks up in map */ }
5
6  void parse(Char *ch, const char *line) {
7    char command[MAX_LINE_LENGTH];
8    char *remainder = one_argument(line, command);
9    DO_FUN *func = find_command(command);
10   if (!func) {
11     send_to_char(ch, "Huh?\n");
12     return;
13   }
14   (*func)(ch, remainder);
15 }
```

PARSING

```
1  typedef void DO_FUN(Char *ch, const char *argument);
2
3  void init_commands() { /* populates some kind of map */ }
4  DO_FUN *find_command(const char *) { /* looks up in map */ }
5
6  void parse(Char *ch, const char *line) {
7    char command[MAX_LINE_LENGTH];
8    char *remainder = one_argument(line, command);
9    DO_FUN *func = find_command(command);
10   if (!func) {
11     send_to_char(ch, "Huh?\n");
12     return;
13   }
14   (*func)(ch, remainder);
15 }
```

PARSING

```
1 // ...
2 void do_save(Char *ch, const char *args);
3 void do_say(Char *ch, const char *args);
4 void do_score(Char *ch, const char *args);
5 void do_sell(Char *ch, const char *arg);
6 void do_set(Char *ch, const char *args);
7 // ...a few hundred more...
```

PARSING

```
1 // ...
2 void do_save(Char *ch);                    // no args!
3 void do_say(Char *ch, ArgParser args);     // takes a parser obj
4 void do_score(Char *ch);                   // no args!
5 void do_sell(Char *ch, ArgParser args);    // takes a parser obj
6 void do_set(Char *ch, const char *arg);    // TODO(#55) update this!
7 // ...a few hundred more...
```

PARSING

```
class ArgParser {
public:
  explicit ArgParser(std::string args);

  bool empty() const;
  std::string_view shift();
  std::optional<int> try_parse_int();
  // other parse methods here...

  // Legacy support...
  const char *c_str() const;
};
```

PARSING

```
1  void add_command(const char *command, DO_FUN *func);
2
3  void init_commands() {
4    // ...
5    add_command("save", do_save);
6    add_command("say", do_say);
7    add_command("'", do_say);
8    add_command("score", do_score);
9    add_command("sell", do_sell);
10   // ... and so on
11 }
```

PARSING

```
1  void add_command(const char *command, DO_FUN *func);
2
3  void init_commands() {
4    // ...
5    add_command("save", do_save);
6    add_command("say", do_say);
7    add_command("'", do_say);
8    add_command("score", do_score);
9    add_command("sell", do_sell);
10   // ... and so on
11 }
```

PARSING

```
1  void add_command(const char *command, DO_FUN *func);
2
3  void init_commands() {
4    // ...
5    add_command("save", do_save);
6    add_command("say", do_say);
7    add_command("'", do_say);
8    add_command("score", do_score);
9    add_command("sell", do_sell);
10   // ... and so on
11 }
```

PARSING

```
 1 void add_command(const char *command, DO_FUN *func);
 2
 3 void init_commands() {
 4   // ...
 5   add_command("save", do_save);
 6   add_command("say", do_say);
 7   add_command("'", do_say);
 8   add_command("score", do_score);
 9   add_command("sell", do_sell);
10   // ... and so on
11 }
```

PARSING

```
using Handler = std::function<void (Char *, ArgParser args)>;

// New-style commands, taking a std::function handler.
void add_command(std::string_view name, Handler handler);

// handle old-style commands of the form do_fun(Char *, const char *)
void add_command(
    std::string_view name, void (*do_fun)(Char *, const char *)) {
  // Add a bespoke lambda to adapt the new call to the old do_fun.
  add_command(
    name,
    [&do_fun](Char *c, ArgParser args) {
      do_fun(c, args.c_str(); }
    });
}
```

PARSING

```
using Handler = std::function<void (Char *, ArgParser args)>;

// New-style commands, taking a std::function handler.
void add_command(std::string_view name, Handler handler);

// handle old-style commands of the form do_fun(Char *, const char *)
void add_command(
    std::string_view name, void (*do_fun)(Char *, const char *)) {
  // Add a bespoke lambda to adapt the new call to the old do_fun.
  add_command(
    name,
    [&do_fun](Char *c, ArgParser args) {
      do_fun(c, args.c_str(); }
    });
}
```

PARSING

```
using Handler = std::function<void (Char *, ArgParser args)>;

// New-style commands, taking a std::function handler.
void add_command(std::string_view name, Handler handler);

// handle old-style commands of the form do_fun(Char *, const char *)
void add_command(
    std::string_view name, void (*do_fun)(Char *, const char *)) {
  // Add a bespoke lambda to adapt the new call to the old do_fun.
  add_command(
    name,
    [&do_fun](Char *c, ArgParser args) {
      do_fun(c, args.c_str(); }
    });
}
```

PARSING

```
using Handler = std::function<void (Char *, ArgParser args)>;

// New-style commands, taking a std::function handler.
void add_command(std::string_view name, Handler handler);

// handle old-style commands of the form do_fun(Char *, const char *)
void add_command(
    std::string_view name, void (*do_fun)(Char *, const char *)) {
  // Add a bespoke lambda to adapt the new call to the old do_fun.
  add_command(
    name,
    [&do_fun](Char *c, ArgParser args) {
      do_fun(c, args.c_str(); }
    });
}
```

PARSING

```
using Handler = std::function<void (Char *, ArgParser args)>;

// New-style commands, taking a std::function handler.
void add_command(std::string_view name, Handler handler);

// handle old-style commands of the form do_fun(Char *, const char *)
void add_command(
    std::string_view name, void (*do_fun)(Char *, const char *)) {
  // Add a bespoke lambda to adapt the new call to the old do_fun.
  add_command(
    name,
    [&do_fun](Char *c, ArgParser args) {
      do_fun(c, args.c_str(); }
    });
}
```