Exercise: srcFacts Concerns Section: Email: Signed Name:

| Allowed Concerns: srcFacts, srcML, XML, Parsing | | |
|---|---|---|
| Concern | srcfacts.cpp#L443-L483 | |
| Parsing | 443 | `// parse start tag` |
| Parsing | 444 | `assert(content.compare(0, "<"sv.size(), "<"sv) == 0);` |
| Parsing | 445 | `content.remove_prefix("<"sv.size());` |
| Parsing | 446 | `if (content[0] == ':') {` |
| Parsing | 447 | `    std::cerr << "parser error : Invalid start tag name\n";` |
| Parsing | 448 | `    return 1;` |
| Parsing | 449 | `}` |
| Parsing | 450 | `std::size_t nameEndPosition = content.find_first_of(NAMEEND);` |
| Parsing | 451 | `if (nameEndPosition == content.size()) {` |
| Parsing | 452 | `    std::cerr << "parser error : Unterminated start tag '" << content.` |
| Parsing | 453 | `    return 1;` |
| Parsing | 454 | `}` |
| Parsing | 455 | `size_t colonPosition = 0;` |
| Parsing | 456 | `if (content[nameEndPosition] == ':') {` |
| Parsing | 457 | `    colonPosition = nameEndPosition;` |
| Parsing | 458 | `    nameEndPosition = content.find_first_of(NAMEEND, nameEndPosition +` |
| Parsing | 459 | `}` |
| XML | 460 | `const std::string_view qName(content.substr(0, nameEndPosition));` |
| Parsing | 461 | `if (qName.empty()) {` |
| Parsing | 462 | `    std::cerr << "parser error: StartTag: invalid element name\n";` |
| Parsing | 463 | `    return 1;` |
| Parsing | 464 | `}` |
| XML | 465 | `[[maybe_unused]] const std::string_view prefix(qName.substr(0, colonPo` |
| XML | 466 | `const std::string_view localName(qName.substr(colonPosition ? colonPos` |
| XML | 467 | `TRACE("START TAG", "qName", qName, "prefix", prefix, "localName", loca` |
| srcML | 468 | `bool inEscape = localName == "escape"sv;` |
| srcML | 469 | `if (localName == "expr"sv) {` |
| srcFacts | 470 | `    ++exprCount;` |
| srcML | 471 | `} else if (localName == "decl"sv) {` |
| srcFacts | 472 | `    ++declCount;` |
| srcML | 473 | `} else if (localName == "comment"sv) {` |
| srcFacts | 474 | `    ++commentCount;` |
| srcML | 475 | `} else if (localName == "function"sv) {` |
| srcFacts | 476 | `    ++functionCount;` |
| srcML | 477 | `} else if (localName == "unit"sv) {` |
| srcFacts | 478 | `    ++unitCount;` |
| srcML | 479 | `} else if (localName == "class"sv) {` |
| srcFacts | 480 | `    ++classCount;` |
| srcML | 481 | `}` |
| Parsing | 482 | `content.remove_prefix(nameEndPosition);` |
| Parsing | 483 | `content.remove_prefix(content.find_first_not_of(WHITESPACE));` |

| Allowed Concerns: srcFacts, srcML, XML, Parsing | |
|---|---|
| **Concern** | **srcfacts.cpp#L293-L302** |
| Parsing | `293  // refill content preserving unprocessed` |
| Parsing | `294  int bytesRead = refillContent(content);` |
| Parsing | `295  if (bytesRead < 0) {` |
| Parsing | `296      std::cerr << "parser error : File input error\n";` |
| Parsing | `297      return 1;` |
| Parsing | `298  }` |
| Parsing | `299  if (bytesRead == 0) {` |
| Parsing | `300      doneReading = true;` |
| Parsing | `301  }` |
| srcFacts | `302  totalBytes += bytesRead` |
| **Concern** | **srcfacts.cpp#L558-L566** |
| Parsing | `558  std::size_t valueEndPosition = content.find(delimiter);` |
| Parsing | `559  if (valueEndPosition == content.npos) {` |
| Parsing | `560      std::cerr << "parser error : attribute " << qName << " missing del` |
| Parsing | `561      return 1;` |
| Parsing | `562  }` |
| XML | `563  const std::string_view value(content.substr(0, valueEndPosition));` |
| srcML | `564  if (localName == "url"sv)` |
| srcFacts | `565      url = value;` |
| XML | `566  TRACE("ATTRIBUTE", "qname", qName, "prefix", prefix, "localName", loca` |
| **Concern** | **srcfacts.cpp#L327-L334** |
| Parsing | `327  // parse character non-entity references` |
| Parsing | `328  assert(content[0] != '<' && content[0] != '&')` |
| Parsing | `329  std::size_t characterEndPosition = content.find_first_of("<&")` |
| XML | `330  const std::string_view characters(content.substr(0, characterEndPositi` |
| XML | `331  TRACE("CHARACTERS", "characters", characters)` |
| srcFacts | `332  loc += static_cast<int>(std::count(characters.cbegin(), characters.cen` |
| srcFacts | `333  textSize += static_cast<int>(characters.size())` |
| Parsing | `334  content.remove_prefix(characters.size())` |