parseTypeAndName

lokale/globale Variablen und Attribute:
begin,end,nameBegin: char*
name,type: Text&
argumentMode: Boolean

skipWhitespaces(begin, end)
skipWhitespacesBackwards(end, begin)
begin = end
wahr: throw runtime_error("Invalid type and name")
falsch: ∅
nameBegin ← end
nameBegin-1 ≠ begin and *(nameBegin-1) ≠ ' '
  nameBegin ← nameBegin - 1
nameBegin-1 = begin
wahr:
  argumentMode
  wahr: type ← string(begin, end)
  falsch: name ← string(begin, end)
  ∅
falsch:
  const char* typeEnd ← nameBegin
  skipWhitespacesBackwards(typeEnd, begin)
  name ← string(nameBegin, end)
  type ← string(begin, typeEnd)

parseOperation

lokale/globale Variablen und Attribute:
name,type: Text
arguments: vector<Argument>
begin,end,retBegin,argBegin: char*
skip: Boolean&
abstract: Boolean
visibility: GZ
argName,argType: Text

match(begin, end, "virtual")
wahr: skipWhitespaces(begin, end)
falsch: ∅
match(begin, end, "explicit")
wahr: skipWhitespaces(begin, end)
falsch: ∅
retBegin ← begin
begin ≠ end and *begin ≠ '('
  begin ← begin + 1
*begin ≠ '('
wahr: throw std::runtime_error("Invalid operation")
falsch: ∅
parseTypeAndName(retBegin, begin, name, type)
begin ← begin + 1
begin = end
wahr: throw std::runtime_error("Invalid operation")
falsch: ∅
skipWhitespaces(begin, end)
*begin ≠ ')'
  argBegin ← begin
  begin ≠ end and *begin ≠ ',' and *begin ≠ ')' and *begin ≠ '='
    begin ← begin + 1
  begin = end
  wahr: throw std::runtime_error("Invalid operation")
  falsch: ∅
  parseTypeAndName(argBegin, begin, argName, argType, true)
  arguments.emplace_back(argName, Type::createFromCppName(argType))
  *begin = '='
  wahr:
    begin ≠ end and *begin ≠ ',' and *begin ≠ ')'
      begin ← begin + 1
  falsch: ∅
  *begin = ','
  wahr: begin ← begin + 1
  falsch: ∅
  skipWhitespaces(begin, end)
begin ← begin + 1
abstract ← false
begin ≠ end
  skipWhitespaces(begin, end)
  match(begin, end, "override")
  wahr: continue
  falsch: ∅
  match(begin, end, "const")
  wahr: continue
  falsch: ∅
  match(begin, end, "=")
  wahr:
    skipWhitespaces(begin, end)
    match(begin, end, "0")
    wahr:
      abstract ← true
      continue
    falsch: ∅
    match(begin, end, "default")
    wahr: continue
    falsch: ∅
    match(begin, end, "delete")
    wahr:
      skip ← true
      continue
    falsch: ∅
  falsch: ∅
  throw runtime_error("Invalid operation")
return Operation(name, Type::createFromCppName(type), arguments, visibility, abstract)