

Manipulace se zdrojovými soubory v C

Úvod

Tento program je vytvářen za účelem zjednodušení a zautomatizování některých úprav zdrojových souborů. Parserem vytvoří syntaktický strom daného zdrojového souboru a podle požadavků ho upraví. Mezi možné úpravy patří:

- přejmenovávání funkcí a proměnných
 - přejmenovávání určité funkce, která se bude rozlišovat podle jména a parametrů, či proměnné, která se bude rozlišovat podle kontextu
- přesunování funkcí a proměnných
 - přesunování funkcí tak, aby se neporušila struktura zdrojového souboru
 - přesunování proměnných do vnějších částí kódu a případné slévání při vícenásobném použití
- vytvoření funkce z části kódu
 - vytvoření funkce z vybrané souvislé části kódu spolu s vytvořením parametrů podle použití lokálních proměnných v tomto kódu
- inline funkce přímo do kódu
 - programové rozbalení volání funkce přímo do místa volání, snaha o úpravu kódu funkce při použití konstant jako parametrů volání
- rozvinutí smyček s konstantním počtem průchodů
 - rozbalení smyček, u kterých lze jednoduše rozhodnout o pevném počtu průchodů, přímo do kódu s použitím konstant místo počítadla

Při všech úpravách se bude brát ohled na správnou strukturu a sémantiku programu.

Výstup programu bude opět do zdrojového souboru.

Tvorba syntaktického stromu je poměrně hodně používána. Využití má například v samotných překladačích, kontrolorech syntaxe, ale i editorů, které jej využívají ke zvýrazňování syntaxe či automatickému doplňování právě psaných konstrukcí.

Některé jednoduché úpravy umožňují provádět i samotné editory, ale většinou jen nad blokem textu, ne nad kusem programu. Optimalizační programy zase naopak provádějí často neprůhledné úpravy, někdy dokonce až na samotném spustitelném souboru na úrovni assembleru. Tento program se pokusí být někde mezi a to v tom smyslu, že bude provádět úpravy, které

uživatel bude chtít, a výstupem bude programátorovi srozumitelný soubor, se kterým bude moci dále pracovat.

Výstup syntaktického stromu bude přehledně formátován. Existuje spousta dalších nástrojů na zpřehlednění a zkrášlení zdrojových souborů.

Struktura programu

Program se bude skládat z několika částí. První bude parser, který vytvoří z preprocesorem zpracovaného zdrojového souboru syntaktický strom. Nad tímto stromem se budou provádět všechny úpravy. Pracovat s ním bude samostatná knihovna, která poskytne bezpečné rozhraní dalším částem programu. Bude mít na starosti také integritu a správnost stromu. Automat bude provádět uživatelem definované změny ve speciálních komentářích přímo ve vstupním zdrojovém souboru. Další možností bude také použití grafického uživatelského rozhraní, ve kterém uživatel bude moci rovnou změny zadávat. Posledním článkem bude export upraveného syntaktického stromu do zdrojového souboru, který už bude stačit pouze přeložit a spustit.

Program bude napsán v c++. Bude mít dvě distribuce – s a bez grafického prostředí. V případě grafického prostředí budou s programem distribuovány běhové knihovny QT. Cílovými platformami budou Linux a Windows.

Reprezentace kódu

Kód programu bude reprezentován v několika provázaných třídách a tabulkách.

Tabulka typů

První úrovní budou globální definice struktur, výčetů a typů. Další úrovní budou tabulky pro každý blok kódu, kde se mohou vyskytnout definice. V této tabulce bude mapován název pojmenovaného typu souboru se základním typem jazyka. Definice typu bude přímo ukazovat na základní typ. Výčet bude rozdělen do jednotlivých konstantních celočíselných typů. Struktura bude rozdělena podle všech svých složek s tečkovanou notací. Typy budou moci samozřejmě odkazovat na již dříve definované. V takovém případě bude základní typ dohledán v tabulce a uložen. Pro struktury obsahující struktury bude tabulka obsahovat s tečkovanou notací i součásti vnořených struktur. Pro struktury obsahující ukazatele na struktury bude tabulka obsahovat s šipkovou notací i součásti vnořených struktur.

Tabulka proměnných

První úrovní budou globální proměnné. Další úrovní budou tabulky pro každý blok kódu, kde se mohou deklarovat proměnné. V této tabulce budou mapovány názvy proměnných s typy z globální tabulky typů.

Tabulka funkcí

V této tabulce budou názvy funkcí a seznam parametrů spolu s jejich typy odkazovat na jednotlivé bloky programu. Budou to vstupní body syntaktického stromu programu.

Třída Statement

Tato třída bude abstraktní a bude popisovat společné rozhraní pro reprezentaci příkazů a klíčových slov jazyka. Budou od ní odvozeny třídy pro konkrétní příkazy podle syntaktického významu. Obsahovat bude ukazatel na výraz, Expression, a pole ukazatelů na příkazy, Statement. Například podtřída If bude používat výraz a jeden nebo dva příkazy pro větve výpočtu, podtřída Return bude používat nanejvýš výraz.

Zvláštními podtřídami budou:

Block - ucelené bloky příkazů. Bude navíc obsahovat ukazatele na tabulku lokálních definic typů a tabulku lokálních proměnných, bude také obsahovat tabulku pojmenování míst mezi příkazy – podle klíčových slov label nebo case.

Expr - pouze výraz nebo prázdný příkaz.

Switch - příkaz s výrazem a posloupností příkazů, navíc bude obsahovat tabulku pojmenování míst mezi příkazy - podle konstantního výrazu case.

Třída Expression

Pomocí této třídy budou reprezentovány veškeré výrazy v programu. Opět to bude abstraktní třída od které budou odvozeny mezitřídě podle počtu operandů. Každý operand bude opět potomek třídy Expression. Od nich teprve třídy pro jednotlivé operátory. Zvláštní třídou bude třída P, která bude pokrývat samostatné identifikátory a unární operátory. Další zvláštní třídou bude třída F pro volání funkce, která bude obsahovat pole ukazatelů na parametry, opět potomky třídy Expression. Volání funkce bude obsahovat index volané funkce v tabulce funkcí pro rychlejší přístup a použití jednotného identifikátoru funkce.

Společné metody

Všechny třídy budou mít virtuální metody eval a print. Metoda eval bude mít za cíl zjednodušení výrazů, případně rozhodovacích příkazů. Jako parametry dostane hodnoty některých proměnných, které pak jako konstanty použije v nižších vrstvách. Metoda print zajistí formátovaný výstup.

Parser

První část programu bude parser. Existuje mnoho parserů jazyka c, případně c++. Po obhlédnutí situace jsem se rozhodl napsat vlastní a to z několika důvodů. Některé mají výstup do poměrně složité struktury syntaktického stromu. Další parsery ke svému běhu používají knihovny dostupné pouze na některé platformě. Navíc pochopení a případné úpravy parserů nejsou jednoduché. Dalším důvodem je procvičení si problematiky psaní parseru. Parser bude napsán přesně na míru, aby dostatečně pokryl požadavky reprezentace kódu a nebyl zbytečně moc složitý a komplexní.

Parser bude jednorůchodový a bude pracovat přímo nad vstupním zdrojovým souborem.

Vstupem bude preprocesorem předzpracovaný zdrojový soubor. Nebude tedy obsahovat makra, konstanty a vkládání dalších souborů. Použitá metoda bude top-down, čili postupné derivování částí kódu. Během práce bude parser postupně vyplňovat tabulky typů a definic typů, funkcí společně s popisem parametrů, struktur a výčtů. Při nalezení syntaktických chyb bude vypsáno chybové hlášení o problému s číslem řádky. V takovém případě program skončí.

Knihovna pro práci se syntaktickým stromem

Syntaktický strom je klíčovou částí programu. Operace nad ním budou vyžadovat přesnost a korektnost. Z tohoto důvodu s ním bude pracovat pouze zvláštní knihovna. Ta se navíc bude skládat ze dvou částí. Takzvané high level a low level. High level knihovna bude poskytovat rozhraní ovládací části programu. Bude podporovat všechny zahrnuté úpravy zdrojového souboru, které bude následně provádět výpočetní úkony a některé optimalizace za pomoci volání funkcí low level knihovny. Bude například správně ovládat funkci eval, kdy bude přebírat návratovou hodnotu a vytvářet nový uzel místo stávajícího.

Low level knihovna bude provádět jednoduché úpravy. Mezi ně bude patřit přejmenování proměnné podle kontextu – od určitého uzlu až po konce nebo dokud nedojde k jejímu zastínění. Bude moci odstraňovat nebo oddělovat uzly stromu či je přepojovat je na jiná místa.

Transformace podle komentářů

Zdrojový soubor bude moci obsahovat speciální komentáře rozpoznávané tímto programem.

Komentáře budou na začátku řádku, budou uvozeny klasickým dvojitým lomítkem // a následovat bude dvojkřížek #. Poté budou následovat klíčová slova. Některá budou uvozovat blok kódu a budou mít ještě párový ukončovací komentář.

Použité komentáře:

- `##rename "name"` - před deklarací funkce či proměnné, v kódu bude přejmenována na řetězec name
- `##inline` - před deklarací funkce, funkce se bude rozbalovat do míst volání
- `##rollout` - před smyčkou, bude se rozbalovat
- `##function_start "name"`
 - začátek kódu, ze kterého bude vytvořena funkce se jménem jako je řetězec name
- `##function_end` - konec kódu, ze kterého bude vytvořena funkce

Tyto úpravy budou prováděny automaticky po vytvoření syntaktického stromu.

GUI

Grafické rozhraní bude naprogramováno pomocí knihoven QT. Jeho běhové knihovny budou distribuovány s programem. Tyto knihovny jsou napsány v c++, což je ve shodě s programem. Výhodou je jednotný zdrojový kód a vzhled pro více platform.

Grafické prostředí bude zobrazovat zdrojový soubor. Nejprve načte vstupní a ten zobrazí. Dále bude moci uživatel zadávat úpravy. Vybírat identifikátory funkcí a proměnných, které bude moci následně stiskem tlačítka zadat k přejmenování. Při vybrání identifikátoru funkce bude moci zadat i možnost rozbalování do míst volání. Bude moci zadat i nové umístění vybrané funkce či proměnné ve zdrojovém souboru. Při vybrání smyčky ji bude moci označit k rozbalení. Při označení kusu kódu, několika příkazů, bude moci zadat jméno funkce, kterou nechá z tohoto místa vytvořit. Po výběru úprav vše jen potvrdí a program upraví podle zadání zdrojový soubor a zobrazí výsledek. Mezi volby bude patřit automatické vyzvání k uložení výstupu a provádění úprav ihned po jejich zadání.

Export

Program bude vypisovat syntaktický strom do zadaného výstupního souboru nebo do grafického prostředí. Nebude měnit vstupní soubor. V této části bude mít uživatel možnost zadat některé konvence formátování, které bude chtít použít.

Například použití tabelátorů nebo pevného počtu mezer. Nastavení počtu mezer kolem operátorů a závorek, případně jen za otevírací a před uzavírací závorkou. Vypisování bloku příkazů vždy do složených závorek i když obsahuje pouze jeden příkaz. Mezi deklaracemi nebo definicemi funkcí vynechání prázdného řádku.

Uživatel bude moci také zadat formát struktury souboru. Například vypsání deklarací typů, výčtů, funkcí a definice struktur na začátek souboru.

Možná rozšíření

Vstup nemusí být předzpracován preprocesorem.

Vyhledávání jen použitých částí ve vkládaných souborech.

Automatické optimalizace kódu – include, rollout, function.

Použití heuristiky v rozhodovacích částech programu.