Implementační dokumentace k 2. úloze do IPP 2023/2024

Jméno a příjmení: Anastasiia Samoilova

Login: xsamoi00

Hlavní práce programu

Po vytvoření třídy `Interpreter`, která je dědicem `Abstract Interpreter` a používá IPP Core, jsou v nástroji Interpreter inicializovány proměnné:

$`vars`: instance třídy `For_Var` pro správu proměnných.

$`labels`: instance třídy `ForLabels` pro správu značek.

$`prog_XML`: XML uzel programu.

$`array_Instr`: instance třídy `Fetcher_for_XML` pro extrakci instrukcí z XML.

$`count`: počítadlo instrukcí, inicializováno nulou.

Po inicializaci proměnných začínáme zpracovávat značky pomocí funkce `Control_Label`:

1. Nastavíme ukazatel na první instrukce v poli instrukcí.

2. Začíná cyklus while, který bude pokračovat, dokud nebude dosaženo konce pole instrukcí. Na každé iteraci je extrahován následující pokyn pomocí metody $`instructions- > NextInstr_for_XML ()`. Zobrazí se hodnota atributu opcode pro aktuální pokyny.

3. Pokud se opcode rovná "LABEL", pak:

   3.1. Argumenty instrukce jsou extrahovány pomocí `Create_Instr:: create Arguments($instructionXml->childNodes)`. Ověřuje se Typ prvního argumentu návodu. Pokud není roven "labelu", vytvoří se chyba.

4. Název značky je odvozen z prvního argumentu. Je povolána metoda `_label_create` objektu $`labels` pro přidání značky do tabulky.

5. Po zpracování aktuálního pokynu se sčítá počítadlo $`count` a přechází na další pokyny.

Funkce `Control_Label` je tedy zodpovědná za zpracování instrukcí s opcode "LABEL", extrahuje jejich argumenty, zkontroluje typ argumentu na "label" a přidá příslušné štítky do tabulky štítků.

Po dokončení se program přesune na začátek cyklu zpracování instrukcí. Spustí se smyčka for, která přeruší každou instrukci z dokumentu XML pomocí objektu `Fetcher_for_XML`.

Pro každý pokyn je povolána metoda, která vytvoří příslušný objekt instrukce pomocí `Create_Instr::create` a provede jej pomocí metody `execute`.

Vytváření instrukcí:

Každá instrukce je extrahována z uzlu XML a je vytvořen objekt instrukce pomocí metody `Create_Instr::create`.

Vytvořené objekty instrukcí jsou prováděny podle jejich typu a argumentů.

Více o mém přístupu při implementaci metod pro práci s pokyny:

Třída `Instruction_Main`:

Tato třída je základní třídou pro všechny pokyny. Obsahuje dvě chráněné metody.

Metoda `typeOperand()` kontroluje Typ operandu. Pokud Typ neodpovídá očekávanému, je vyhozena chyba s kódem 32.

Metoda `ValueOperand()` kontroluje hodnotu operandu. Pokud hodnota neodpovídá tomu, co se očekávalo, je vyhozena chyba s kódem chyby 53.

Každý návod je definován svou třídou, která se dědí z Instruction_Main.

Každá třída pokynů implementuje metodu `execute()`, která provádí odpovídající pokyny.Každá třída pokynů přijímá argumenty a objekty pro práci s proměnnými a štítky, které jsou nezbytné pro provádění pokynů.

V rámci metody `execute()` každé instrukce jsou prováděny konkrétní akce podle jejího účelu.

Pro pohodlnější práci jsem rozdělila základní pokyny do několika skupin: Logic sekce(jsou zpracovávány ve třídě `Logic_Instr`(to je AND, OR, NOT), aritmetika (ADD, SUB, MUL, DIV) atd. to pomohlo udržet strukturovanější kód.

Tento kód tedy umožňuje dynamické vytváření a provádění instrukcí založených na popisu XML, kontrolu typů a hodnot operandů a správu toku provádění programu.

Provádění pokynů:

V metodě `execute` dochází k cyklickému provádění každé instrukce z dokumentu XML.

Každá instrukce je zpracována pomocí metody `process_for_instr`, která provádí příslušný objekt instrukce.

Program tedy nejprve inicializuje všechny potřebné datové struktury, poté zpracuje štítky a poté provede pokyny obsažené v dokumentu XML.

Paměťový model:

Koncepty rámců, proměnných a štítků se používají k organizaci paměti a řízení provádění programu. Pojďme rozebrat každý z těchto aspektů na základě poskytnutého kódu:

Práce s proměnnými (třída `For_Var`):

Třída `For_Var` představuje práci s proměnnými a rámci. Poskytuje metody k určení, nastavení a získání hodnot proměnných.

Uvnitř třídy jsou uloženy různé typy rámců: globální (GF), dočasný (TF) a místní (LF).

Metoda `var_define`() se používá k určení nové proměnné. Rozdělí řetězec zobrazení typ @ název na typ a název proměnné a poté zkontroluje, zda proměnná již nebyla definována v aktuálním rámečku.

Metody `var_set()` a `var_get()` se používají k nastavení a získání hodnot proměnných podle jejich názvu a typu.

Práce s rámy:

Třída `For_Var` poskytuje metody pro práci s rámci: vytváření, přidávání a mazání.

Metody `frame_create()`, `frame_push()` a `frame_pop()` jsou navrženy tak, aby vytvořily nový dočasný rámeček, přidaly jej do zásobníku místních rámců a odstranily Poslední rámeček ze zásobníku.

V závislosti na typu rámečku (GF, TF, LF) používaného metodou `get_frame()` dochází k návratu příslušného slovníku obsahujícího proměnné uvnitř rámce.

Práce s labely (třída `ForLabels`):

Třída `ForLabels` je zodpovědná za správu značek v programu.

Metoda `_label_create()` se používá k vytvoření nového štítku se zadaným názvem a pozicí v programu.

Metoda `label_get_index()` vrací pozici (index) značky podle jejího názvu.

Metody `get_address_safe()` a `return_address()` se používají k bezpečnému ukládání a extrahování návratových adres.

Tyto třídy tak umožňují pohodlný přístup k proměnným a štítkům a také správu rámce, což umožňuje efektivní provádění pokynů.

Závěr

K testování projektu byly použity testy, které nám byly poskytnuty z oficiálního zdroje. Níže můžete vidět UML diagram, který ukazuje graficky můj projekt.