Методи на Транслация

Входен ЕП

доц. д-р Александър Пенев

Входен ЕП

Входен ЕП

Под **Входен Език за Програмиране** ще разбираме ЕП, на който е записана входната програма, която транслатора анализира и преобразува в изходна програма.

За описанието на Входния ЕП обикновено разглеждаме:

- Данни (логически, синтактичен, реализационен);
- Операции (логически, синтактичен, реализационен);
- Управление на последователността от действия;
- Управление на данни;
- Управление на паметта;
- Операционна среда;

Данни във Входният ЕП

(логически, синтактичен и реализационен аспекти)

Входен ЕП – Данни

- Логически аспект – Абстрактната дефиниция на типа данни, например: Масив се нарича множество от еднотипни величини, между чиито имена е установена строга наредба посредством биективно изображение в свързано подмножество на Z;
- Синтактичен аспект – Дефиницията на масив позволява използването на цели числа, като индекси на масива, които идентифицират елементите му:

```
var A1: array[2..10] of byte;

type Color = (Red, Green, Blue, White);
var A2: array[Red..Blue] of byte;
```

- Реализационен аспект – определя начините за представяне на данните по време на изпълнение на програмата;

Входен ЕП – Данни

По време на изпълнение на програмата съществуват два класа данни:

- Данни определени от програмиста;
- Данни от системата (транслатора) – те се формират от реализацията на езика т.е. от транслатора за служебни нужди и често програмиста не подозира за тях. Една от причините за съществуването им е късното свързване. Например, ако входния език позволява в хода на изпълнение на програмата една променлива да получава стойности от различен тип, то по време на изпълнение на програмата ще трябва да се съхранява не само стойността, но и описание на типа и в момента. Такъв системен тип данни се нарича дескриптор на данни;

Входен ЕП – Представяне на данни

Под представяне на данни в ОП се разбира двойката от следните два елемента:

- Код – двоично число, представящо текущата стойност на величината;
- Дескриптор – код, съдържащ допълнителна информация за първия елемент от представянето (т.е. за кода). Тази информация може да бъде името на данната, типът и (т.е. начинът на обработка на кода и), местоположението и размерът по време на изпълнение и др.;

Очевидно всяко представяне на данни:

- Може да не съдържа дескриптор, но е задължително да има код;
- Може да бъде в последователни клетки в ОП или да е „разхв.“;

Входен ЕП – Видове операции над данни

Очевидно че всяка операция над данна получава достъп до нея чрез адреса на позицията на данната.

Една операция наричаме:

- Специфична – ако алгоритъмът и не използва дескриптора от позициите на нейните аргументи и резултат;
- Универсална – в противен случай;

Въпрос: Какво са операциите във бВИМ?

Входен ЕП – Видове операции над данни

Декларация ще наричаме всяка инструкция на ЕП, която описва характеристиките на данните по време на изпълнение. Следователно на всяка декларация съответства позиция от паметта по време на изпълнение.

Пример: `var A: array[2..10] of byte;`

- Име на величината: A;
- Тип на величината: масив;
- Тип на елемента: byte;
- Брой на величините: 9;
- Множество от индексите: 2..10; и др.

Очевидно декларациите позволяват универсалните операции на входния ни ЕП да се преобразуват в специфични във бВИМ;

Входен ЕП – Видове данни

Различаваме два вида данни:

- Прости;
- Съставни – достъп до елемент на съставен тип данни:
  - Достъп до стойност: `X = A[5] + 1;`
  - Достъп до адрес: `A[5] = X + 1;`

Операции във Входният ЕП

(логически, синтактичен и реализационен аспекти)

Входен ЕП – Операции

- Логически аспект – Операцията се разглежда като алгоритъм над абстрактни типове данни. Например алгоритъм за изчисляване на корен квадратен от X;
- Синтактичен аспект – Начинът на описание на операциите в ЕП. Има два начина: като знак в програмата (т.е. вграден в ЕП, което означава че програмиста не е длъжен да знае алгоритъма) или като подпрограма (т.е. описана с алгоритъм, който програмиста трябва да знае):

```
A+B, sin(x), ...
float f(float X) { return X + X - 1; }, ...
```

- Реализационен аспект – Начинът на описание на алгоритъмът в термините на бВИМ;

Входен ЕП – Операции

Операциите в ЕП могат да се разделят на три класа:

- Вградени – тези които се означават със знак и има съответствие в бВИМ.
  Например: `A + B;`
- Примитивни – тези които се означават със знак и се реализират като последователности от операции в бВИМ.
  Например: `sin(X);`
- Подпрограми;

Входен ЕП – Обзор на Операциите

Елементарни:

- Аритметични операции;
- Операции за сравнение (предикати);
- Логически: конюнкция, дизюнкция, отрицание;
- Операции по преобразуване на типове (явни и неявни);

Неелементарни:

- Присвояване;
- Създаване на структура от данни;
- Добавяне на елемент в структура то данни;
- Унищожаване на структура от данни;
- Премахване на елемент от структура от данни;
- Подпрограми;

Управление на Последователността от Действия във Входният ЕП

(операции в изрази, оператори, подпрограми)

Входен ЕП – Управление последователността

- Механизъм за управление на последователността в изрази в ЕП – основен механизъм е функционалната композиция;
- Механизъм за управлението на последователността на оператори в ЕП – всеки използван такъв механизъм трябва да бъде преводим в термините на бВИМ (неявен механизъм за последователно изпълнение, условен и безусловен преход, ...). Може да се използват етикети и явни оператори за преход или оператори за разклонение и цикъл;
- Механизъм за управление на последователността на изпълнение на подпрограми – обикновено в бВИМ има команди за преход към подпрограма (call, gosub) и възврат (ret);

Входен ЕП – Управление посл. на подпрограми

Механизъм за управление на последователността на подпрограми се базира на т.нар. правило за копиране: *Ефектът от изпълнение на оператор за извикване на подпрограма е същият както, ако преди извикването си е заменен от копие на тялото на подпрограмата, в която са направени съответните замени на формалните с фактическите параметри.*

Това правило обикновено има следните ограничение:

- Подпрограмите не могат да бъдат рекурсивни;
- Необходим е явен оператор за извикване на подпрограма;
- При всяко извикване подпрограмата се изпълнява до логическия си край;
- Последователността на изпълнение на подпрограмите е единствена (т.е. в даден момент се изпълнява една подпрог.);
- Незабавно предаване на управлението в точката на възврат;

Управление на Данните във Входният ЕП

Входен ЕП – Управление на данните

- Всяка програма изпълнява алгоритмите си конкретни фиксирани данни, а над множества допустими стойности;
- Поради това позоваванията на данни в преобладаващата им част трябва да бъдат косвени т.е. да се използват имена (идентификатори);
- Във ЕП от високо ниво се позволява едно има да се използва повече от един път т.е. в различни моменти от времето на изпълнение с него да се означават различни програмни елементи, както и един и един и същ обект да се именува с различни имена е един и същ момент от времето;
- Основно място заема операцията позоваване на данни т.е. с какво е асоцииран даден идентификатор в даден момент;

Входен ЕП – УД – асоциация на идентификатор

- Асоциация ще наричаме наредената двойка (идентификатор, дескриптор на данни)
- В транслаторите тези асоциации се записват в т.нар. **Таблица на символите**;

Идентификатор	Дескриптор
i	variable, int, ...
f	function, float, ...
...	...

Входен ЕП – УД операции

Основните операции за УД, които имат отношение към асоц. са:

- Именуване – означава създаване на асоциация;
- Разименуване – унищожаване на асоциация т.е. идент. вече не идентифицира, това което е било асоциирано с него до момента;
- Активиране на асоциация – разрешение за ползване на асоциация от операцията позоваване;
- Деактивиране на асоциацията – разрешението за ползване се променя на „забранено“ или „неактивно“;
- Обработка на позоваване – търсене в таблицата на имената сред активните асоциации;

Множеството от активни в момента асоциации ще наричаме **среда на позоваване**. Съвкупността от първите 4 операции ще обозначаваме като **област на действие** (на име);

Входен ЕП – УД области на действие

В зависимост от това дали тези правила отразяват процеса на изпълнение или транслация на програмата се различават съответно:

- Динамична област на действие;
- Статична област на действие;

От гледна точка на блочната структура на блочните ЕП имаме:

- Локална среда за позоваване – изградена от локални асоциации;
- Нелокална среда за позоваване – изградена от глобални и нелокални асоциации;

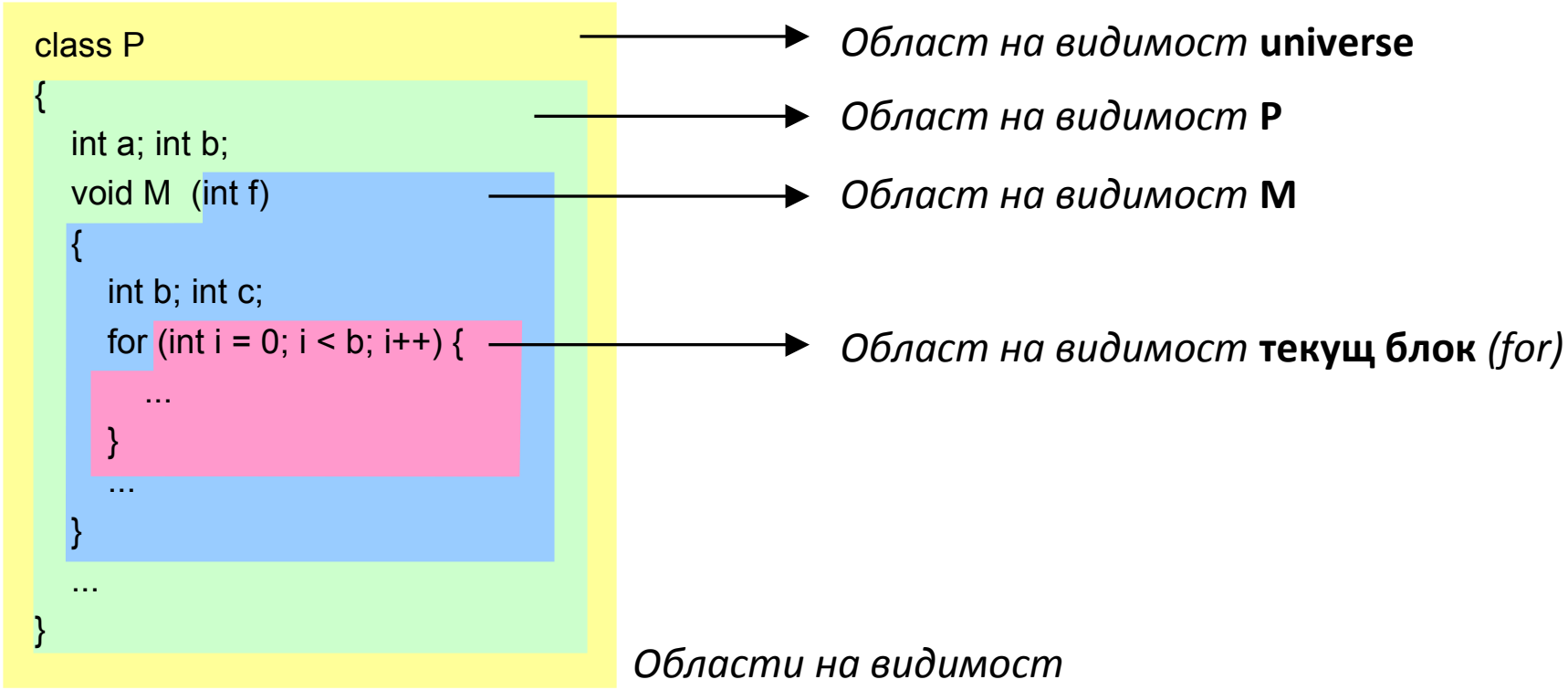
Входен ЕП – УД области на действие

Изводи:

- Областта на действие произтича от механизма за управление на последователността на действия (УПД) на подпрограми;
- Динамичната област на действие предполага наличие и използване на среда за позоваване в изпълнимата програма;
- Практически областта на действие се фиксира строго (най-вече в частта операция търсене) от разработчика, но отсъства в явен вид в ръководствата на езика;

Входен ЕП – Област на видимост (Scope)

Област на видимост на име е региона, в който името е валидно



Управление на Паметта във Входният ЕП

Входен ЕП – Управление на паметта

- Управление на паметта е една от основните грижи, както на програмиста, така и на разработчика на транслатори;
- Основните обекти, за които е необходима памет по време на изпълнение са:
  - Транслираната програма;
  - Програмите по време на изпълнение;
  - Дефинираните от програмата данни;
  - Памет за механизма на УПД (точки на възврат и др.);
  - Среда на позоваване;
  - Временна памет за междинните резултати при изчисляване на изрази;
  - Временната памет за предаване на параметри на подпр.;
  - Буфери за В/И, системни данни и др.

Входен ЕП – Управление на паметта

Съществуват три основни аспекта при УП:

- Разпределение (начално) на паметта;
- Утилизация на паметта – откриване на заделена, но неизползвана памет;
- Уплътнение на паметта – събиране на неизползваните части на едно последователно място (може и на повече от едно);

Операционна Среда

ЕП – Операционна среда

- Операционна среда на дадена програма, това е множеството от програми с която тя комуникира посредством съобщения;
- Това може да са: ОС; други програми в системата; същата програма, но на други компютърни системи (разпределени приложения) и др.

Въпроси?

apenev@uni-plovdiv.bg