

парадигмы разные нужны,
парадигмы разные важны

ЛЕКЦИЯ №3

ALGEBRAIC PROGRAMMING

SYSTEM APS

доц. Песчаненко В.С.

Херсонский государственный университет

Кафедра информатики

Научно-исследовательский институт информационных технологий

Лаборатория по разработке и внедрению педагогических программных средств

2010

ALGEBRAIC PROGRAMMING SYSTEM APS

Области применения:

- прототипирование алгоритмов,
- верификация,
- доказательство программ,
- компьютерная алгебра,
- алгебра логики,
- конвертация данных,
- парсеры

Если данные можно представить в виде дерева, и это дерево надо как-то преобразовывать — в этом случае точно можно использовать APS.



ALGEBRAIC PROGRAMMING SYSTEM APS WHAT ARE NEW?

- Язык **FPL** (**F**ormula **P**rocessing **L**anguage) – позволяет использовать язык **APLAN** на C++.
- Интеграция с технологиями [SmartPtr](#) Интеграция с технологиями SmartPtr, [GMP library](#).
- Реализация процедуры хеширования в алгоритме переписывания.
- Реализация функций добавления, удаления и обновления правил системы переписывающих правил (задача хеширования термов).
- Реализация недетерминированной стратегии переписывания.



ALGEBRAIC PROGRAMMING SYSTEM APS

Файловая структура:

- `aps.exe`
- `intlib.dll`
- `intlib_can.tbl`
- `intlib_proc.tbl`
- `sublib.dll`
- `sublib_proc.tbl`
- `usrlib.dll`
- `usrlib_can.tbl`
- `usrlib_proc.tbl`



ALGEBRAIC PROGRAMMING SYSTEM APS

_proc.tbl (описание C++ процедур):

- 1 строка: 30 – количество записей в файле
- Оставшиеся строки имеют вид:

<APLAN имя> v[V][S]... <C++ имя> <имя библиотеки>

V – интерпретировать

S – копировать

Количество больших **V,S** – *количество параметров процедуры*

□ **Пример:**

let vVS let

В языке APLAN C++ функцию *let* можно вызвать используя имя *let* с двумя параметрами. Первый параметр интерпретируется, второй – копируется.



ALGEBRAIC PROGRAMMING SYSTEM APS

Определения:

- ▣ *Канонизатор* – C++ функция или система переписывающих правил которая строит каноническую форму.
- ▣ *Отметка* – тип данных APS, который определяется так:

MARK <имя>(<арность>[,<приоритет>[,<>[,<0 или 1
для арности 2>]]])

и.ф.з для арности 2

и.ф.з – инфиксная форма записи.

0 – дерево для одинаковых вершин – правосторонние,
а 1 – левосторонние

Пример: ADD (2, 54, "+"). $ADD(x,y) \Leftrightarrow x+y$



ALGEBRAIC PROGRAMMING SYSTEM APS

_can.tbl (описание C++ канонизаторов отметок):

- 1 строка – количество записей в файле
- Оставшиеся строки имеют вид:

<имя> <имя и.ф.з> <C++ имя> <имя библиотеки>

□ **Пример:**

ADD add add_can

ADD – имя отметки, которое должно быть определено (std.ap:
ADD(2,...))

add – имя введено специально для процедуры markcan (1 add 2)

add_can – имя C++ функции



ALGEBRAIC PROGRAMMING SYSTEM APS

- *Основная структура данных* – правосторонние дерево

Типы данных APS:

- Имена,
- Отметки,
- Целые, вещественные числа, строки,
- АТОМЫ

Совет:

Если написанная программа - не работает и Вы нашли место в котором ошибка – *проверьте приоритеты операций*



ALGEBRAIC PROGRAMMING SYSTEM APS

Имена:

- Могут быть глобальные или локальные.
- Доступны только с момента объявления.
- Локальные имена доступны только до конца файла в котором они объявлены

□ Объявление глобальных имен:

```
NAME <имя> [“[“<нат. число>”]”];
```

□ Объявление локальных имен:

```
PRIVATE <имя> [“[“<нат. число>”]”];
```



ALGEBRAIC PROGRAMMING SYSTEM APS

▣ Примеры:

NAME T;

NAME P[3];

P[1]:=P[3];

PRIVATE P;



ALGEBRAIC PROGRAMMING SYSTEM APS

- **Целые числа** – GMP library
- **Вещественные числа** – GMP library, разделитель «.»
- **Строки** – как в C++
- **Скобки:** [], {}.
- **Атомы:**

Если это ни один из перечисленных выше типов, тогда это АТОМ.

- **Явное объявление:**

АТОМ <имя>;

- Локальные и формальные параметры процедур, считаются именами с областью видимости – тело процедуры.
- Если имена пересекаются с локальными или формальными параметрами процедуры, то из тела процедуры не будет доступа к глобальному имени.



ALGEBRAIC PROGRAMMING SYSTEM APS

□ 1 столбик `_proc.tbl` – **АТОМЫ**

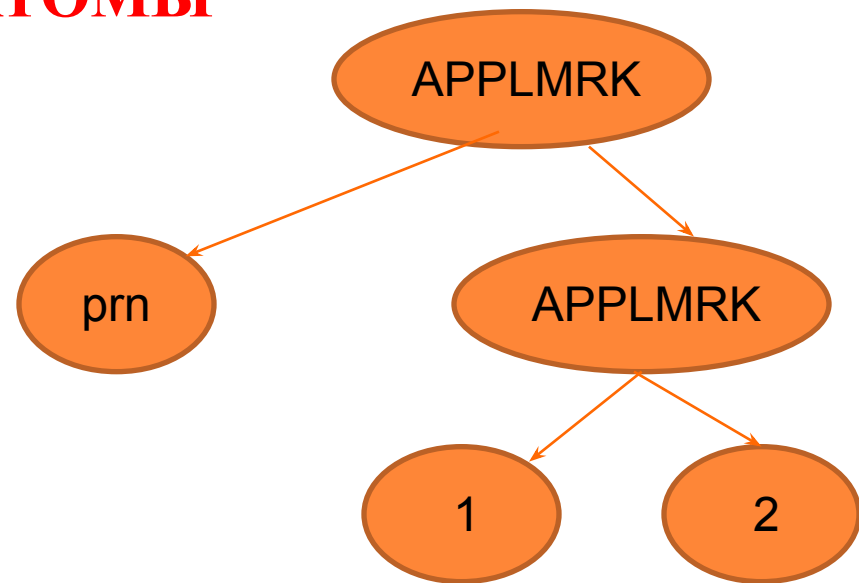
□ **Примеры:**

`prn 1 2` \Rightarrow 1 2

`prn(1 2)` \Rightarrow 1 2

`prn 1;2` \Rightarrow 1

`prn(1;2)` \Rightarrow 1;2



□ **APPLMRK** – отметка с наивысшим приоритетом в APS.

□ **Круглые скобочки** в APS влияют на построение дерева, но в самом дереве они отсутствуют.



ALGEBRAIC PROGRAMMING SYSTEM APS

□ 1 столбик `_can.tbl` – имена ОТМЕТОК

`prn 1+2 => 1`

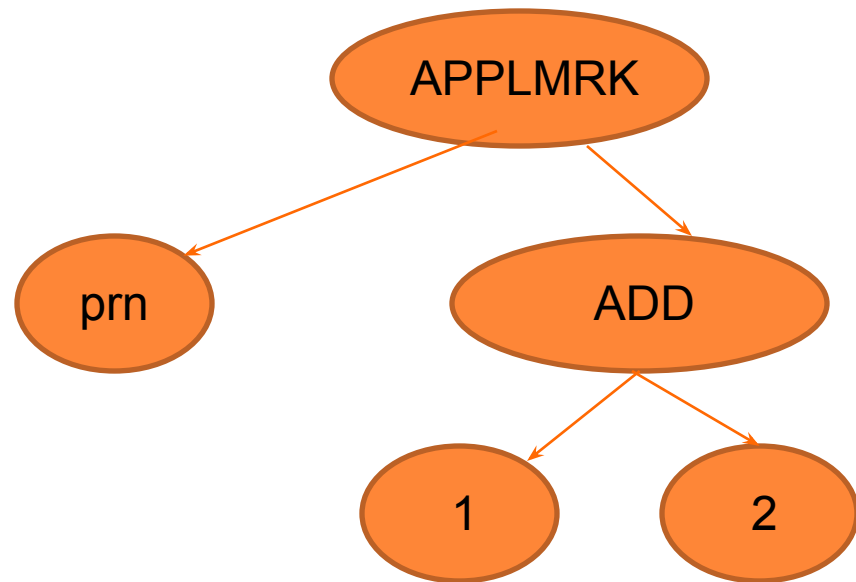
`prn(1+2) =>3`

□ **can_tbl.tbl:**

`ADD add add_can`

□ **std.ap:**

`ADD(2, 54, "+")`



ALGEBRAIC PROGRAMMING SYSTEM APS

▣ Пример канонизатора на APLAN:

```
INCLUDE <std.ap>
```

```
INCLUDE <gen_obj.ap>
```

```
MARK myADD(2,54,"+!");
```

```
NAME R;
```

```
R:=rs(x,y)(
```

```
  x+!y=x+y
```

```
);
```

```
task:= markcan(0+!0,R),prn (1+!2);
```

Попробуйте сделать канонизатор для GCD



ALGEBRAIC PROGRAMMING SYSTEM APS

Файл `std.ap`:

- Определение основных отметок для выражений и операторов
- Определение точки входа в программу.
- Если не подключить `std.ap`, то канонизаторы отметок описанные в `can_tbl.tbl` работать не будут (Например `+` - будет атомом, а не инфиксной формой записи некой отметки).
- **Минимальная программа APS:**

```
NAME task;
```

```
task:=prn "Hello world";
```



ALGEBRAIC PROGRAMMING SYSTEM APS

Файл `gen_obj.ap` – дополнительные определения отметок + `compile`.

Правило интерпретации оператора APS:

- Вызов процедуры, описанной в `proc_tbl.tbl`
- Вызов канонизатора, если он установлен для верхней отметки дерева
- В противном случае APS вызывает систему переписывающих правил `compile`
- Если оператора нет и в `compile` – сообщение об ошибке и продолжение работы.
- **Расширение языков возможностей APS проходит в `compile`**
- **Подключение `gen_obj.ap` существенно упрощает процесс разработки на APS**



ALGEBRAIC PROGRAMMING SYSTEM APS

- ▣ **Стратегия** – некая процедура, которая реализует некий обход дерева с применением системы переписывающих правил.
- ▣ **Файл strat.ap** –определения различных стратегий стратегий(некоторые из них в комментарии, т.к. перенесены на C++, можно их откомментировать – тогда они будут использоваться. **Почему?**)
- ▣ **Процесс разработки на APS:**

Прототип на APLAN => анализ прототипа и его усовершенствование => реализация C++



ALGEBRAIC PROGRAMMING SYSTEM APS

Джентльменский набор файлов APS:

- `aps.exe`
- `intlib.dll`
- `intlib_can.tbl`
- `intlib_proc.tbl`
- `sublib.dll`
- `sublib_proc.tbl`
- `usrlib.dll`
- `usrlib_can.tbl`
- `usrlib_proc.tbl`
- `std.ap`
- `gen_obj.ap`
- `strat.ap`

