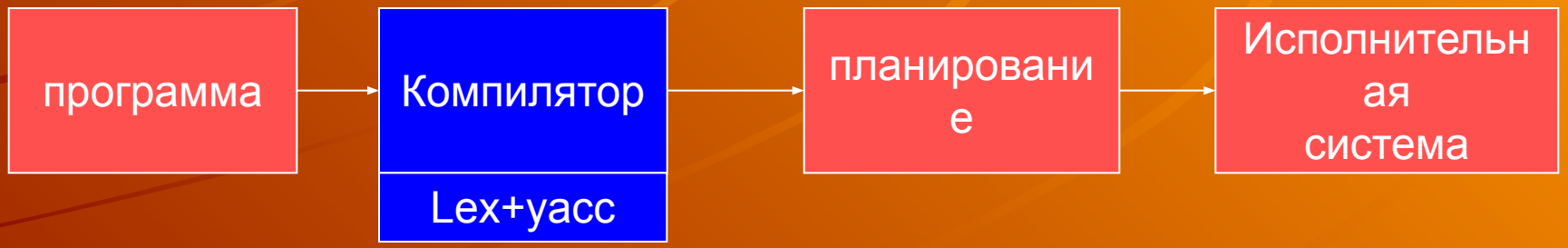
Усовершенствование языка и компилятора

Для системы фрагментированного программирования

Крупин Сергей ФИТ НГУ 3 курс
Руководитель: Перепёлкин Владислав

Летняя Школа по параллельному программированию 2010 г.

Цели и задачи



- Необходимо было создать удобную среду разработки программ, модифицировав уже существующую.
- Для лексического и синтаксического анализа использовались утилиты Bison и Lex.

Старый и новый синтаксис

```
df x(100);
df y_1(100);
df y_2(100);
df y_3(100);
df y_4(100);
cf a = func(out: x);
cf b_1 = norm(in x;out:y_1);
cf b_2 = norm(in x;out:y_2);
cf b_3 = norm(in x;out:y_3);
cf b_4 = norm(in x;out:y_4);
b_1 < a;
b_2 < a;
b_3 < a;
b_4 < a;
```

```
const N = 10*10;
const M = 2;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M*M;)norm(in:x; out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Для описания реальных задач требуется описывать множества фрагментов данных и вычислений

Были добавлены:

- Константы
- Разбор выражений из констант и итераторов
- Массивы
- Циклические описания фрагментов вычислений
- Циклическое задание порядка исполнения фрагментов вычислений

Константы

```
const N = 100;
```

```
const N = 10*10;
const M = 2;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M*M;)norm(in:x;
out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Разбор выражений из констант и итераторов

```
const N = 10*10;
const M = 2;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M*M;)norm(in:x;
out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Массивы

```
const N = 10*10;
const M = 4;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M;)norm(in:x; out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Циклические описания фрагментов вычислений

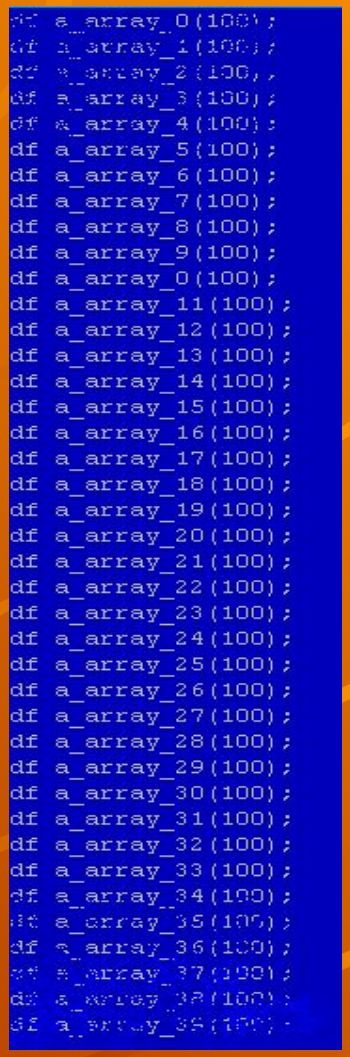
```
const N = 10*10;
const M = 4;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M;)norm(in:x; out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Циклическое задание порядка исполнения фрагментов вычислений

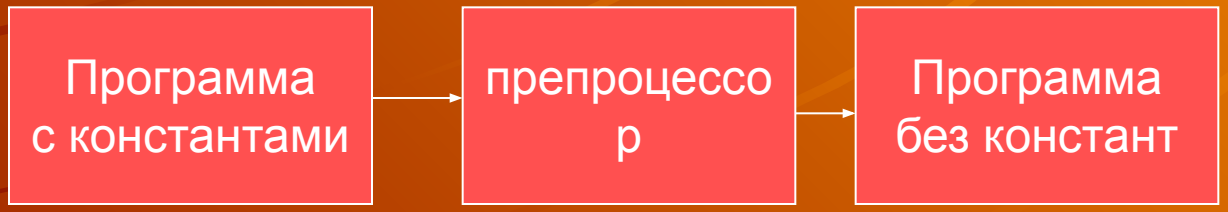
```
const N = 10*10;
const M = 4;
df x(N);
ar[M] df y(N);
cf a = func(out:x);
cf b = for(i=1..M;)norm(in:x; out:y[i-1]);
b[i-1] < a (i=1..4;);
```

Способ реализации I (неэффективный)

- Был добавлен препроцессор который разворачивал циклы. Довольно неплохой способ за исключением того, что генерировался огромный объем кода.



Способ реализации II (эффективный)



Были изменены функции препроцессора – теперь он стал только подставлять константы и упрощать по возможности выражения. Циклы передаются в систему в свернутом виде.

```
        fs->SymbolTable[102] = "A";
        for(int i = 0; i < 25; i++)
{
        fp.DFs[102+i].Size = 12800;
}
        fs->SymbolTable[127] = "B";
        for(int i = 0; i < 25; i++)
{
        fp.DFs[127+i].Size = 12800;
}
        fs->SymbolTable[152] = "C";
        for(int i = 0; i < 25; i++)
{
        fp.DFs[152+i].Size = 12800;
}
        fs->SymbolTable[177] = "Ctmp";
        for(int i = 0; i < 125; i++)
{
        fp.DFs[177+i].Size = 12800;
}
        fs->SymbolTable[302] = "mult";
        fp.CodeFrags[302] = mult;
for(int i = 0; i <= 4; i++ ){
for(int j = 0; j <= 4; j++ ){
for(int k = 0; k <= 4; k++ ){
        fs->SymbolTable[303] = "M";
        fp.CFs[303+(((0*(4 - 0 + 1)+i)*
        fp.CFs[303].InputDFs.push_back(
        fp.CFs[303].InputDFs.push_back(
        fp.CFs[303].OutputDFs.push_back
}}}
        fs->SymbolTable[428] = "sum";
        fp.CodeFrags[428] = sum;
for(int i = 0; i <= 4; i++ ){
for(int j = 0; j <= 4; j++ ){
for(int k = 0; k <= 4; k++ ){
        fs->SymbolTable[429] = "S";
        fp.CFs[429+(((0*(4 - 0 + 1)+i)*
        fp.CFs[429].InputDFs.push_back(
        fp.CFs[429].InputDFs.push_back(
        fp.CFs[429].OutputDFs.push_back
}}}
```

Итог

- Был разработан, протестирован и интегрирован в исходную систему удобный язык для разработки фрагментированных программ.