Динамические структуры данных

Мингалёв Олег

Московский государственный университет им. М. В. Ломоносова,

Факультет вычислительной математики и кибернетики,

101 группа

oleg@mingalev.net

10 марта 2014 г.

Оглавление

Глава 1

Постановка задачи

1.1 Формулировка задачи

Заданы два многочлена, найдите их сумму.

1.2 Формат входных данных

Многочленом считается алгебраическая сумма одночленов вида `aX^n`, `aX`, `X^n` и `a`, завершённая переводом строки, причём $a > 0$, $n > 1$. Нулевой многочлен задаётся строкой `0\n`.

Формально, `<poly> ::= ([-]([<num>]X[^<num>]|<num>)((+|-)([<num>]X[^<num>]|<num>)*)|(0))\n`.

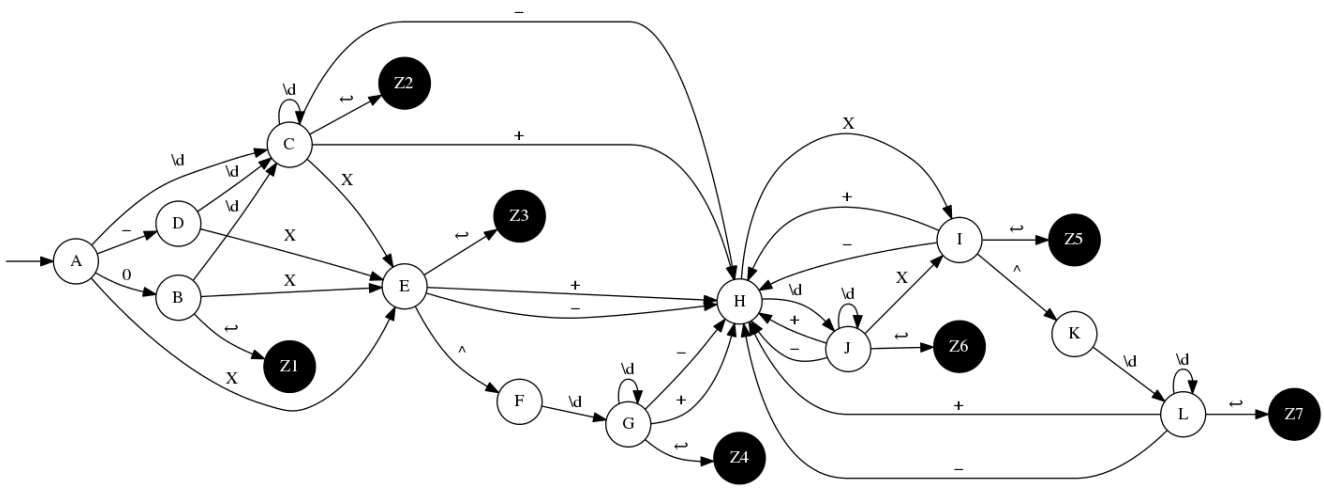
Одночлены одной степени не повторяются.

1.3 Формат выходных данных

Необходимо вывести сумму двух введённых многочленов в формате, описанном в разделе «Формат входных данных» по убыванию степеней одночленов.

Глава 2

Лексемный анализ



Синтаксический разбор многочленов представляет из себя моделирование вышепредставленного автомата.

Выделенные вершины — терминальные.

Глава 3

Тестирование

```
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^2+X+1
2nd poly: X^2+X-1
Sum: 2X^2+2X
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^2+2X
2nd poly: -X^2-3X+10
Sum: -X+10
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^12+16X
2nd poly: 0
Sum: X^12+16X
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: 0
2nd poly: 0
Sum: 0
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: -X^3+X^2+1
2nd poly: -1-X^2+X^3
Sum: 0
```

```
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: +X^2+X
Syntax error at position 1: Symbols {-0123456789X} excepted but '+'[43] found
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^2+X^1
Error: Powers should be at least 2
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: 2X^3+3X+23X^3
Error: At least two monominals with power 3
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: 0X+2
Error: Multipliers must be nonzero
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^X
Syntax error at position 3: Symbols {0123456789} excepted but 'X'[88] found
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: 12*X
Syntax error at position 3: Symbols {\n0123456789X+-} excepted but '*'[42] found
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: XX^2
Syntax error at position 2: Symbols {\n^+-} excepted but 'X'[88] found
[shhdup@shhdup-think lab02]$ ./lab02
lab02 v0.0.4, Mingalev Oleg 2014

Use lab02 -h to see help page

Variant: 16
Sum of two polynomials
==============================
1st poly: X^2+
Syntax error at position 5: Symbols {X0123456789} excepted but new line found
```

Приложение А

Исходный код

Makefile

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include<unistd.h>

struct node {
    int a, k;
    struct node *next;
};

void error(const char *msg) {
    printf("Error: %s\n", msg);
    exit(1);
}

void syntax_error(int pos, char c, const char *valid) {
    if (c != 13 && c != 10) printf("Syntax error at position %d: Symbols {%s} excepted but '%c'[%d
        ] found\n", pos, valid, c, c);
    else printf("Syntax error at position %d: Symbols {%s} excepted but new line found\n", pos,
        valid);
```

```
20      exit(1);
21  }
22
23  struct node* lstcpy(struct node *list) {
24      if (!list) return 0;
25      struct node *foo = (struct node *) malloc(sizeof(struct node));
26      foo->a = list->a;
27      foo->k = list->k;
28      foo->next = 0;
29      return foo;
30
31  }
32
33  char TERM = '\n';
34
35  struct node* insert(struct node *list, int k, int a) {
36      if (a == 0) {
37          static const char *err = "Multipliers must be nonzero";
38          error(err);
39      }
40      if (!list) {
41          struct node *foo = (struct node *) malloc(sizeof(struct node));
42          foo->a = a;
43          foo->k = k;
44          foo->next = 0;
45          return foo;
46      }
47      if (k > list->k) {
48          struct node *foo = (struct node *) malloc(sizeof(struct node));
49          foo->a = a;
50          foo->k = k;
51          foo->next = list;
52          return foo;
53      } else if (k == list->k) {
54          char err[100];
55          sprintf(err, "At least two monominals with power %d", k);
```

```
56          error(err);
57      } else {
58          list->next = insert(list->next, k, a);
59          return list;
60      }
61  }
62
63  void free_list(struct node *list) {
64      if (!list) return;
65      free_list(list->next);
66      free(list);
67  }
68
69  struct node* scan(void) {
70      struct node *list = 0;
71      char state = 'A';
72      char c;
73      int a; int sgn; int k;
74      int pos = 0;
75      while (state != 'Z') {
76          c = getc(stdin);
77          ++pos;
78          switch (state) {
79              case 'A': {
80                  static const char *valid = "-0123456789X";
81                  sgn = 1; k = 0; a = 0;
82                  if (c == '0') {
83                      state = 'B';
84                      break;
85                  }
86                  if (c == '-') {
87                      sgn = -1;
88                      state = 'D';
89                      break;
90                  }
91                  if (c == 'X') {
```

```
                    a = 1; k = 1;
                    state = 'E';
                    break;
                }
                if (isdigit(c)) {
                    state = 'C';
                    a = c - '0';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'B': {
                static const char *valid = "\\n0123456789X";
                if (c == TERM) {
                    state = 'Z';
                    break;
                }
                if (isdigit(c)) {
                    a = c - '0';
                    state = 'C';
                    break;
                }
                if (c == 'X') {
                    a = 0; k = 0;
                    state = 'E';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'C': {
                static const char *valid = "\\n0123456789X+-";
                if (c == TERM) {
                    list = insert(list, 0, sgn*a);
                    state = 'Z';
                    break;
                }
```

```
                if (c == 'X') {
                    k = 1;
                    state = 'E';
                    break;
                }
                if (isdigit(c)) {
                    a = 10*a + c - '0';
                    break;
                }
                if (c == '+') {
                    list = insert(list, 0, sgn*a);
                    state = 'H';
                    sgn = 1;
                    break;
                }
                if (c == '-') {
                    list = insert(list, 0, sgn*a);
                    state = 'H';
                    sgn = -1;
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'D': {
                static const char *valid = "\\nX";
                if (isdigit(c)) {
                    a = c - '0';
                    state = 'C';
                    break;
                }
                if (c == 'X') {
                    state = 'E';
                    k = 1; a = 1;
                    break;
                }
                syntax_error(pos, c, valid);
```

```
            }
            case 'E': {
                static const char *valid = "\\n^+-";
                if (c == TERM) {
                    list = insert(list, 1, sgn*a);
                    state = 'Z';
                    break;
                }
                if (c == '^') {
                    state = 'F';
                    break;
                }
                if (c == '+') {
                    list = insert(list, 1, sgn*a);
                    sgn = 1;
                    state = 'H';
                    break;
                }
                if (c == '-') {
                    list = insert(list, 1, sgn*a);
                    sgn = -1;
                    state = 'H';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'F': {
                static const char *valid = "0123456789";
                if (isdigit(c)) {
                    k = c - '0';
                    state = 'G';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'G': {
```

```
                static const char *valid = "\\n0123456789+-";
                if (isdigit(c)) {
                    k = 10*k + c - '0';
                    break;
                }
                if (c == TERM) {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, sgn*a);
                    state = 'Z';
                    break;
                }
                if (c == '+') {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, sgn*a);
                    sgn = 1;
                    state = 'H';
                    break;
                }
                if (c == '-') {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, sgn*a);
                    sgn = -1;
                    state = 'H';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'H': {
                static const char *valid = "X0123456789";
                k = 0; a = 0;
                if (c == 'X') {
                    state = 'I';
                    a = 1;
                    break;
                }
                if (isdigit(c)) {
```

```
                    state = 'J';
                    a = c - '0';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'I': {
                static const char *valid = "\\n+-^";
                if (c == TERM) {
                    list = insert(list, 1, sgn*a);
                    state = 'Z';
                    break;
                }
                if (c == '+') {
                    list = insert(list, 1, sgn*a);
                    sgn = 1;
                    state = 'H';
                    break;
                }
                if (c == '-') {
                    list = insert(list, 1, sgn*a);
                    sgn = -1;
                    state = 'H';
                    break;
                }
                if (c == '^') {
                    state = 'K';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'J': {
                static const char *valid = "\\nX0123456789+-";
                if (isdigit(c)) {
                    a = a*10 + c - '0';
                    break;
```

```
                }
                if (c == 'X') {
                    state = 'I';
                    k = 1;
                    break;
                }
                if (c == TERM) {
                    state = 'Z';
                    list = insert(list, 0, sgn*a);
                    break;
                }
                if (c == '+') {
                    list = insert(list, 0, sgn*a);
                    sgn = 1;
                    state = 'H';
                    break;
                }
                if (c == '-') {
                    list = insert(list, 0, sgn*a);
                    sgn = -1;
                    state = 'H';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'K': {
                static const char *valid = "0123456789";
                if (isdigit(c)) {
                    k = c - '0';
                    state = 'L';
                    break;
                }
                syntax_error(pos, c, valid);
            }
            case 'L': {
                static const char *valid = "\\n+-0123456789";
```

```
                if (c == TERM) {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, a*sgn);
                    state = 'Z';
                    break;
                }
                if (isdigit(c)) {
                    k = 10*k + c - '0';
                    break;
                }
                if (c == '+') {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, a*sgn);
                    state = 'H';
                    sgn = 1;
                    break;
                }
                if (c == '-') {
                    if (k < 2) error((char *)"Powers should be at least 2");
                    list = insert(list, k, a*sgn);
                    state = 'H';
                    sgn = -1;
                    break;
                }
                syntax_error(pos, c, valid);
            }
        }
    }
    return list;
}

struct node* print(struct node *list) {
    struct node *root = list;
    if (!list) {
        printf("0\n"); return 0;
    }
```

```
344     int first = 1;
345     while (list) {
346         if (list->a > 0) {
347             if (!first) printf("+");
348             if (list->a > 1 || list->k == 0) printf("%d", list->a);
349         }
350         if (list->a < 0) {
351             printf("-");
352             if (list->a < -1 || list->k == 0) printf("%d", -list->a);
353         }
354         if (list->k > 0) printf("X");
355         if (list->k > 1) printf("^%d", list->k);
356         list = list->next;
357         first = 0;
358     }
359     printf("\n");
360     return root;
361 }
362 
363 struct node* sum(struct node *l1, struct node *l2) {
364     struct node *ans = 0;
365     if (!l1 && !l2) return 0;
366     if (!l1) return insert(sum(l1, l2->next), l2->k, l2->a);
367     if (!l2) return insert(sum(l1->next, l2), l1->k, l1->a);
368     if (!l2) return l1;
369     if (l1->k > l2->k) return insert(sum(l1->next, l2), l1->k, l1->a);
370     if (l1->k < l2->k) return insert(sum(l2->next, l1), l2->k, l2->a);
371     if (l1->a + l2->a)
372         return insert(sum(l1->next, l2->next), l1->k, l1->a + l2->a);
373     else
374         return sum(l1->next, l2->next);
375 }
376 
377 void variant_info(void) {
378     printf("lab02␣v0.0.4,␣Mingalev␣Oleg␣2014\n\n");
379     printf("Use␣lab02␣-h␣to␣see␣help␣page\n\n");
```

```
380     printf("Variant: 16\n");
381     printf("Sum of two polynomials\n");
382     printf("================================\n");
383 }
384 
385 void help(void) {
386     printf("lab02 v0.0.4, Mingalev Oleg 2014\n");
387     printf("Usage: lab02 [-h]\n\n");
388     printf("Polynomial format:");
389     printf("<expr> ::= [+|-]<mono>((+|-)<mono>)*\n");
390     printf("<mono> ::= <int> | [<int>]X[^<int>]\n");
391     printf("<int>  ::= (0..9)+\n");
392     printf("All multipliers must be nonzero\n");
393     printf("All powers should be at least 2\n");
394     printf("To specify zero polynomial just enter 0");
395     exit(0);
396 }
397 
398 int main(int argc, char *argv[]) {
399     variant_info();
400     char cur;
401     while ((cur = getopt(argc, argv, "hrtv")) != -1) {
402         switch (cur) {
403             case 'h': case '?': help(); break;
404         }
405     }
406     printf("1st poly: "); fflush(stdout); struct node *l1 = scan();
407     printf("2nd poly: "); fflush(stdout); struct node *l2 = scan();
408     struct node *ls = sum(l1, l2);
409     printf("Sum: "); print(ls);
410     free_list(l1); free_list(l2); free_list(ls);
411     return 0;
412 }
```