

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, $\langle \text{poly} \rangle ::= ([-] ([\langle \text{num} \rangle] X^{[\langle \text{num} \rangle]} | \langle \text{num} \rangle) ((+|-) ([\langle \text{num} \rangle] X^{[\langle \text{num} \rangle]} | \langle \text{num} \rangle) *) | (0)) \backslash n$.

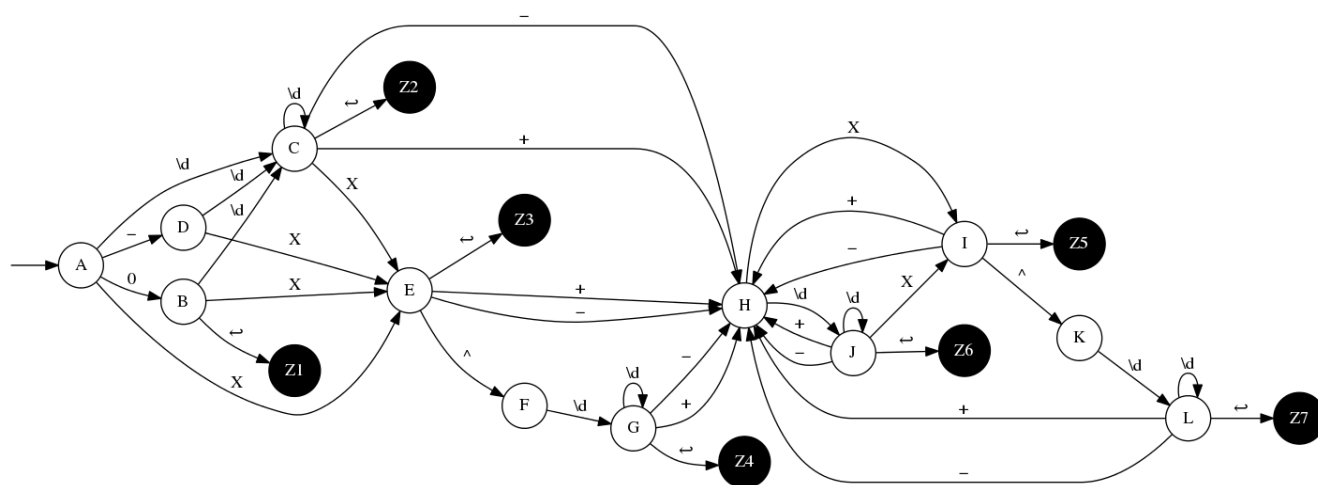
Одночлены одной степени не повторяются.

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<ctype.h>
4 #include<string.h>
5 #include<unistd.h>
6
7 struct node {
8     int a, k;
9     struct node *next;
10 };
11
12 void error(const char *msg) {
13     printf("Error:_%s\n", msg);
14     exit(1);
15 }
16
17 void syntax_error(int pos, char c, const char *valid) {
18     if (c != 13 && c != 10) printf("Syntax_error_at_position_%d:_%Symbols_{%s}_excepted_but_'%c'[%d
19         ]_found\n", pos, valid, c, c);
20     else printf("Syntax_error_at_position_%d:_%Symbols_{%s}_excepted_but_new_line_found\n", pos,
21         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 monomials 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') {

```

```

92         a = 1; k = 1;
93         state = 'E';
94         break;
95     }
96     if (isdigit(c)) {
97         state = 'C';
98         a = c - '0';
99         break;
100    }
101    syntax_error(pos, c, valid);
102}
103case 'B': {
104    static const char *valid = "\\n0123456789X";
105    if (c == TERM) {
106        state = 'Z';
107        break;
108    }
109    if (isdigit(c)) {
110        a = c - '0';
111        state = 'C';
112        break;
113    }
114    if (c == 'X') {
115        a = 0; k = 0;
116        state = 'E';
117        break;
118    }
119    syntax_error(pos, c, valid);
120}
121case 'C': {
122    static const char *valid = "\\n0123456789X+ -";
123    if (c == TERM) {
124        list = insert(list, 0, sgn*a);
125        state = 'Z';
126        break;
127    }

```

```

128         if (c == 'X') {
129             k = 1;
130             state = 'E';
131             break;
132         }
133         if (isdigit(c)) {
134             a = 10*a + c - '0';
135             break;
136         }
137         if (c == '+') {
138             list = insert(list, 0, sgn*a);
139             state = 'H';
140             sgn = 1;
141             break;
142         }
143         if (c == '-') {
144             list = insert(list, 0, sgn*a);
145             state = 'H';
146             sgn = -1;
147             break;
148         }
149         syntax_error(pos, c, valid);
150     }
151     case 'D': {
152         static const char *valid = "\\nX";
153         if (isdigit(c)) {
154             a = c - '0';
155             state = 'C';
156             break;
157         }
158         if (c == 'X') {
159             state = 'E';
160             k = 1; a = 1;
161             break;
162         }
163         syntax_error(pos, c, valid);

```

```

164     }
165     case 'E': {
166         static const char *valid = "\\n^+-";
167         if (c == TERM) {
168             list = insert(list, 1, sgn*a);
169             state = 'Z';
170             break;
171         }
172         if (c == '^') {
173             state = 'F';
174             break;
175         }
176         if (c == '+') {
177             list = insert(list, 1, sgn*a);
178             sgn = 1;
179             state = 'H';
180             break;
181         }
182         if (c == '-') {
183             list = insert(list, 1, sgn*a);
184             sgn = -1;
185             state = 'H';
186             break;
187         }
188         syntax_error(pos, c, valid);
189     }
190     case 'F': {
191         static const char *valid = "0123456789";
192         if (isdigit(c)) {
193             k = c - '0';
194             state = 'G';
195             break;
196         }
197         syntax_error(pos, c, valid);
198     }
199     case 'G': {

```

```

200     static const char *valid = "\\n0123456789+-";
201     if (isdigit(c)) {
202         k = 10*k + c - '0';
203         break;
204     }
205     if (c == TERM) {
206         if (k < 2) error((char *)"Powers should be at least 2");
207         list = insert(list, k, sgn*a);
208         state = 'Z';
209         break;
210     }
211     if (c == '+') {
212         if (k < 2) error((char *)"Powers should be at least 2");
213         list = insert(list, k, sgn*a);
214         sgn = 1;
215         state = 'H';
216         break;
217     }
218     if (c == '-') {
219         if (k < 2) error((char *)"Powers should be at least 2");
220         list = insert(list, k, sgn*a);
221         sgn = -1;
222         state = 'H';
223         break;
224     }
225     syntax_error(pos, c, valid);
226 }
227 case 'H': {
228     static const char *valid = "X0123456789";
229     k = 0; a = 0;
230     if (c == 'X') {
231         state = 'I';
232         a = 1;
233         break;
234     }
235     if (isdigit(c)) {

```

```

236         state = 'J';
237         a = c - '0';
238         break;
239     }
240     syntax_error(pos, c, valid);
241 }
242 case 'I': {
243     static const char *valid = "\\n+-^";
244     if (c == TERM) {
245         list = insert(list, 1, sgn*a);
246         state = 'Z';
247         break;
248     }
249     if (c == '+') {
250         list = insert(list, 1, sgn*a);
251         sgn = 1;
252         state = 'H';
253         break;
254     }
255     if (c == '-') {
256         list = insert(list, 1, sgn*a);
257         sgn = -1;
258         state = 'H';
259         break;
260     }
261     if (c == '^') {
262         state = 'K';
263         break;
264     }
265     syntax_error(pos, c, valid);
266 }
267 case 'J': {
268     static const char *valid = "\\nX0123456789+ -";
269     if (isdigit(c)) {
270         a = a*10 + c - '0';
271         break;

```

```

272     }
273     if (c == 'X') {
274         state = 'I';
275         k = 1;
276         break;
277     }
278     if (c == TERM) {
279         state = 'Z';
280         list = insert(list, 0, sgn*a);
281         break;
282     }
283     if (c == '+') {
284         list = insert(list, 0, sgn*a);
285         sgn = 1;
286         state = 'H';
287         break;
288     }
289     if (c == '-') {
290         list = insert(list, 0, sgn*a);
291         sgn = -1;
292         state = 'H';
293         break;
294     }
295     syntax_error(pos, c, valid);
296 }
297 case 'K': {
298     static const char *valid = "0123456789";
299     if (isdigit(c)) {
300         k = c - '0';
301         state = 'L';
302         break;
303     }
304     syntax_error(pos, c, valid);
305 }
306 case 'L': {
307     static const char *valid = "\\n+-0123456789";

```

```

308         if (c == TERM) {
309             if (k < 2) error((char *)"Powers should be at least 2");
310             list = insert(list, k, a*sgn);
311             state = 'Z';
312             break;
313         }
314         if (isdigit(c)) {
315             k = 10*k + c - '0';
316             break;
317         }
318         if (c == '+') {
319             if (k < 2) error((char *)"Powers should be at least 2");
320             list = insert(list, k, a*sgn);
321             state = 'H';
322             sgn = 1;
323             break;
324         }
325         if (c == '-') {
326             if (k < 2) error((char *)"Powers should be at least 2");
327             list = insert(list, k, a*sgn);
328             state = 'H';
329             sgn = -1;
330             break;
331         }
332         syntax_error(pos, c, valid);
333     }
334 }
335 }
336 return list;
337 }
338
339 struct node* print(struct node *list) {
340     struct node *root = list;
341     if (!list) {
342         printf("0\n"); return 0;
343     }

```



```

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: %16s\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 }

```