BÁO CÁO ĐỀ TÀI KẾT THÚC HỌC PHẦN

BỘ MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Lớp: 09

STT Nhóm: 21

Nguyễn Thanh Nam – 21110904

Đặng Thế Kỷ - 21110893

Lê Minh Hoàng – 21110457

TÊN ĐỀ TÀI: VIẾT CHƯƠNG TRÌNH MÔ PHỎNG DẠNG HÌNH ẢNH ĐỂ BIỂU DIỄN ĐA THỨC DƯỚI DẠNG DANH SÁCH LIÊN KẾT VÀ HỖ TRỢ PHÉP TOÁN CỘNG, NHÂN ĐA THỨC

1. PHÂN TÍCH THUẬT TOÁN
   a. *Thuật toán cộng hai đa thức*

Algorithm A (*Addition of polynomials*). This algorithm adds polynomial(P) to polynomial(Q), assuming that P and Q are pointer variables pointing to polynomials having the form above. The list P will be unchanged; the list Q will retain the sum. Pointer variables P and Q return to their starting points at the conclusion of this algorithm; auxiliary pointer variables Q1 and Q2 are also used.

A1. [Initialize.] Set P ← LINK(P), Q1 ← Q, Q ← LINK(Q). (Now both P and Q point to the leading terms of their polynomials. Throughout most of this algorithm the variable Q1 will be one step behind Q, in the sense that Q = LINK(Q1).)

A2. [ABC(P):ABC(Q).] If ABC(P) < ABC(Q), set Q1 ← Q and Q ← LINK(Q) and repeat this step. If ABC(P) = ABC(Q), go to step A3. If ABC(P) > ABC(Q), go to step A5.

A3. [Add coefficients.] (We've found terms with equal exponents.) If ABC(P) < 0, the algorithm terminates. Otherwise set COEF(Q) ← COEF(Q) + COEF(P). Now if COEF(Q) = 0, go to A4; otherwise, set P ← LINK(P), Q1 ← Q, Q ← LINK(Q), and go to A2. (Curiously the latter operations are identical to step A1.)

A4. [Delete zero term.] Set Q2 ← Q, LINK(Q1) ← Q ← LINK(Q), and AVAIL ⇐ Q2. (A zero term created in step A3 has been removed from polynomial(Q).) Set P ← LINK(P) and go back to A2.

A5. [Insert new term.] (Polynomial(P) contains a term that is not present in polynomial(Q), so we insert it in polynomial(Q).) Set Q2 ⇐ AVAIL, COEF(Q2) ← COEF(P), ABC(Q2) ← ABC(P), LINK(Q2) ← Q, LINK(Q1) ← Q2, Q1 ← Q2, P ← LINK(P), and return to step A2. ∎

INPUT: hai đa thức P và Q với các hạng tử được sắp xếp theo chiều giảm dần tổng số mũ.

OUTPUT: đa thức mới là kết quả của phép cộng hai đa thức P và Q.

FINITENESS: khởi tạo một hạng tử có lũy thừa âm cho mỗi đa thức P và Q. Trong quá trình duyệt, nếu duyệt đến hạng tử có lũy thừa âm của đa thức thì thuật toán sẽ dừng lại. Điều đó cho thấy sự hữu hạn của thuật toán phép công đa thức.

DEFINITENESS:

ABC(P): Lũy Thừa của P

ABC(Q): Lũy Thừa của Q

COEF(P): Hệ số của P

COEF(Q): Hệ số của Q

Thuật toán tường minh, các bước đều rõ ràng, không có gì mơ hồ hoặc không thể xác định.

EFFECTIVENESS: Khoảng O(n+m) với n là số phần tử của P, m là số phần tử của Q.

b. *Thuật toán nhân hai đa thức*

Algorithm B (*Multiple of polynomials*).

B1. [Initialize.] Set Q ← LINK(Q), Res ← 0, If ABC(Q) < 0, the algorithm terminates.

B2. [Multiply.] Multiply each term of Q by the polynomial P and save the result in the variable temp, then Res ← Res + temp (using Algorithm A). Return to step B1.

INPUT: hai đa thức P và Q với các hạng tử được sắp xếp theo chiều giảm dần tổng số mũ.

OUTPUT: đa thức mới là kết quả của phép nhân hai đa thức P và Q.

FINITENESS: ban đầu thêm 1 hạng tử có số mũ âm cho cả 2 đa thức. Khi duyệt tới hạng tử có số mũ âm thì kết thúc vòng lặp (if(ABC(Q) < 0, the algorithm terminates).

DEFINITENESS: thuật toán tường minh, các bước đều rõ ràng, không có gì mơ hồ hoặc không thể xác định.

EFFECTIVENESS: Khoảng O(n*(n+m)) với n là số phần tử của P, m là số phần tử của Q.

2. CODE

a. *Tạo struct Node và viết hàm lưu hạng tử của đa thức vào circular list*

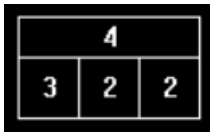
```
 7  struct Node {
 8      int coeff;
 9      int powx;
10      int powy;
11      int powz;
12      struct Node* next;
13  };
14  void create_node(int c, int p1, int p2, int p3, struct Node** temp)
15  {
16      Node* r = (Node*)malloc(sizeof(Node)); // Stores new node
17      Node* z = *temp; // Stores temp node
18      r->coeff = c; // Update coefficient of r
19      r->powx = p1; // Update power of variable x in r
20      r->powy = p2; // Update power of variable y in r
21      r->powz = p3; // Update power of variable z in r
22      if (z == nullptr) {
23          (*temp) = r; // Update temp node
24          (*temp)->next = (*temp); // Update next pointer of temp node
25      }
26      else {
27          r->next = z->next; // Update next pointer of z
28          z->next = r; // Update next pointer of z
29          *temp = r; // Update temp Node
30      }
31  }
```

b. Vẽ một hạng tử của đa thức

```
32  void draw_Node(int left, int top, int right, int bottom, Node* node) {
33      if (node->coeff == 0) return;
34      char* numberstring = (char*)malloc(sizeof(char));
35      sprintf(numberstring, "%d", node->coeff);
36      outtextxy(left + 42, top + 2, numberstring);
37      sprintf(numberstring, "%d", node->powx);
38      outtextxy(left + 12, bottom + 7, numberstring);
39      sprintf(numberstring, "%d", node->powy);
40      outtextxy(left + 42, bottom + 7, numberstring);
41      sprintf(numberstring, "%d", node->powz);
42      outtextxy(left + 72, bottom + 7, numberstring);
43      rectangle(left     , top     , right, bottom);
44      rectangle(left     , bottom, right, bottom + 30);
45      rectangle(left + 30, bottom, right, bottom + 30);
46      rectangle(left + 60, bottom, right, bottom + 30);
47  }
```

Dùng hàm vẽ *rectangle*(int left, int top, int right, int bottom) để tạo thành một ô node và chia ô thành các phần coefficient(hệ số), powx(lũy thừa của x), powy(lũy thừa của y), powz(lũy thừa của z).

- Dùng hàm *outtextxy*(int x, int y, char *textstring) để in các giá trị ra màn hình.
- Ví dụ hạng tử $4x^3y^2z^2$:



c. Vẽ đa thức

```
48  void draw_Poly(int left, int top, int right, int bottom, Node* node) {
49      int l = left;
50      int r = right;
51      Node* temp = node;
52      node = node->next;
53      while (node != temp) {
54          if (node->coeff != 0) {
55              draw_Node(l, top, r, bottom, node);
56              l += 100;
57              r += 100;
58              if (r >= getmaxwidth()) {
59                  l = left;
60                  r = right;
61                  top += 70;
62                  bottom += 70;
63              }
64          }
65          node = node->next;
66          delay(100);
67      }
68      if (node->coeff != 0) draw_Node(l, top, r, bottom, node);
69  }
```

- Dùng vòng lặp while lần lượt duyệt và in các hạng tử có hệ số khác 0 của đa thức ra màn hình.
- Ví dụ đa thức $4x^3y^2z^2 + 5x^2yz^4 + 3xyz^3 + 2xyz^2 + 2xz^4 + 3yz^3 + 2z$:

4			5			3			2			2			3			2		
3	2	2	2	1	4	1	1	3	1	1	2	1	0	4	0	1	3	0	0	1

d. Biểu diễn đa thức ở màn hình console

```
71  void display(struct Node* node)
72  {
73      Node* start = node; // Stores head node of list
74      Node* temp = (Node*)malloc(sizeof(Node));
75      node = node->next; // Update node
76      while (node != start && node->coeff != 0) {
77
78          cout << node->coeff; // Print coefficient of current node
79
80          if (node->powx != 0) cout << "x^" << node->powx;
81          if (node->powx != 0 && node->powy != 0) cout << "*";
82          if (node->powy != 0) cout << "y^" << node->powy;
83          if ((node->powy != 0 && node->powz != 0) || (node->powx != 0 && node->powz != 0)) cout << "*";
84          if (node->powz != 0) cout << "z^" << node->powz;
85
86          if (node != start && node->next->coeff != 0) // Add next term of the polynomial
87              cout << " + ";
88
89          node = node->next; // Update node
90      }
91      if (node->coeff != 0) {
92          cout << node->coeff;
93          if (node->powx != 0) cout << "x^" << node->powx;
94          if (node->powx != 0 && node->powy != 0) cout << "*";
95          if (node->powy != 0) cout << "y^" << node->powy;
96          if ((node->powy != 0 && node->powz != 0) || (node->powx != 0 && node->powz != 0)) cout << "*";
97          if (node->powz != 0) cout << "z^" << node->powz;
98      }
99      cout << "\n\n";
100 }
```

e. Tạo chuyển động cho hình ảnh trong cửa sổ đồ họa

- Sử dụng hàm *getimage*(int left, int top, int right, int bottom, void *bitmap) để lấy ảnh trên cửa sổ đồ họa bằng phép XOR (vẽ trên màn hình chỗ nào đã có màu lần trước thì màu đó được in đè lên do ta sử dụng xor).
- Sử dụng hàm *bar*(int left, int top, int right, int bottom) kết hợp với *setcolor*(int color) để xóa hình ảnh tại khu vực đã lấy ảnh.
- Sử dụng vòng lặp *while* kết hợp với *putimage*(int left, int top, void *bitmap, int op) để tạo chuyển động cho ảnh đã sao chép. Sử dụng hàm *putimage* lần thứ nhất để dán hình ảnh đã cắt tại điểm chỉ định, sử dụng lần thứ hai để xóa hình ảnh đó (do hình ảnh lần trước được in ra, sau đó in thêm một lần nữa, theo phép xor thì 1 xor 1 = 0 tức là 2 hình ảnh đó đè lên nhau sẽ ra màu nền cũ trước đó).

f. Cộng hai đa thức (Algorithm A kết hợp vẽ bằng thư viện graphics.h)

- Tạo cửa sổ đồ họa, biểu diễn hai đa thức:

```
110     initwindow(getmaxwidth(), getmaxheight());
111     draw_Poly(10, 10, 100, 30, P);
112     delay(500);
113     draw_Poly(10, 80, 100, 100, Q);
114     delay(500);
```

- Bước khởi tạo:

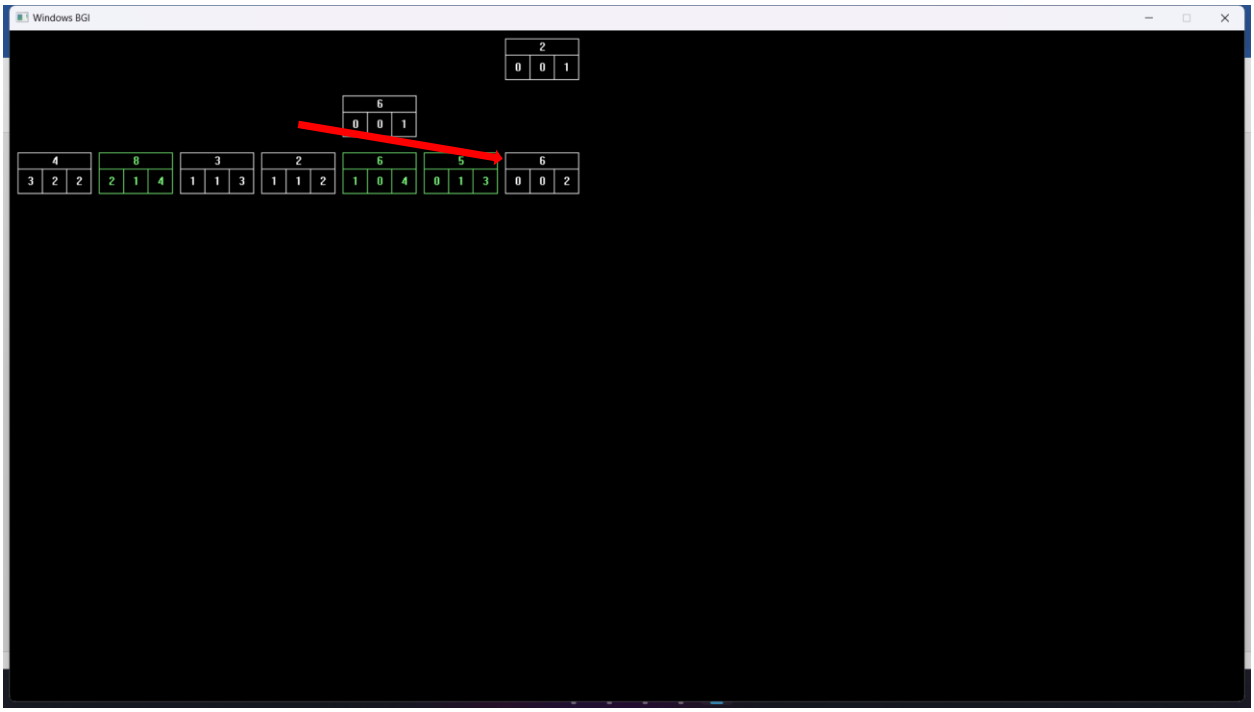
```
123     P = P->next;
124     Node* Q1 = Q;
125     Q = Q->next;
```

- Hàm so sánh lũy thừa hai hạng tự của hai đa thức:

```
101 int checkABC(Node* N1, Node* N2) {
102     if (N1->powx == N2->powx && N1->powy == N2->powy && N1->powz == N2->powz) return 0;
103     if ((N1->powx < N2->powx)
104         || (N1->powx == N2->powx && N1->powy < N2->powy)
105         || (N1->powx == N2->powx && N1->powy == N2->powy && N1->powz < N2->powz)) return -1;
106     return 1;
107 }
```

- Nếu ABC(P) < ABC(Q):

```
132     if (check == -1) {
133         Q1 = Q;
134         create_node(Q1->coeff, Q1->powx, Q1->powy, Q1->powz, &Res);
135         Q = Q->next;
136         char* p;
137         int n = imagesize(0, 0, 100, 60);
138         p = (char*)malloc(n);
139         getimage(x2, y2, x2 + 90, y2 + 51, p);
140         float x = x2, y = y2, tempX = 0, tempY = 0;
141         float addX = abs(x - xRes) / 100, addY = abs(y - 150) / 100;
142         bar(x2, y2, x2 + 91, y2 + 51);
143         do {
144             putimage(x + tempX, y + tempY, p, 1);
145             delay(5);
146             putimage(x + tempX, y + tempY, p, 1);
147             tempX += addX;
148             tempY += addY;
149         } while (y + tempY < 150);
150         putimage(x + tempX, y + tempY, p, 1);
151         delete(p);
152         x2 += 100;
153         xRes += 100;
154         delay(5);
155     }
```



- Nếu ABC(P) = ABC(Q):

```
156         else if (check == 0) {
157             if (P->powz < 0) {
158                 char* p;
159                 int n = imagesize(0, 0, 1200, 60);
160                 p = (char*)malloc(n);
161                 getimage(10, 150, xRes, 210, p);
162                 bar(10, 150, xRes, 210);
163                 float temp = 0;
164                 do {
165                     putimage(10, 150 - temp, p, 1);
166                     delay(5);
167                     putimage(10, 150 - temp, p, 1);
168                     temp += 1.4;
169                 } while (150 - temp > 10);
170                 putimage(10, 150 - temp, p, 1);
171                 delete(p);
172                 display(Res);
173                 getch();
174                 closegraph();
175                 return;
176             }
177             Q->coeff += P->coeff;
178             if (Q->coeff == 0) {
179                 Q2 = Q;
180                 Q1->next = Q = Q->next;
181                 create_node(Q1->coeff, Q1->powx, Q1->powy, Q1->powz, &Res);
182                 P = P->next;
183             }
```

```
184             else {
185                 P = P->next;
186                 Q1 = Q;
187                 create_node(Q1->coeff, Q1->powx, Q1->powy, Q1->powz, &Res);
188                 Q = Q->next;
189                 char* p;
190                 int n = imagesize(0, 0, 100, 60);
191                 p = (char*)malloc(n);
192                 getimage(x1, y1, x1 + 90, y1 + 50, p);
193                 float x = x1, y = y1, tempX = 0, tempY = 0;
194                 float addX = (x - x2) / 100, addY = abs(y - y2) / 100;
195                 bar(x1, y1, x1 + 91, y1 + 51);
196                 do {
197                     putimage(x + tempX, y + tempY, p, 1);
198                     delay(5);
199                     putimage(x + tempX, y + tempY, p, 1);
200                     if (x1 > x2) tempX -= addX;
201                     else tempX += addX;
202                     tempY += addY;
203                 } while (y + tempY < y2);
204                 putimage(x + tempX, y + tempY, p, 1);
205                 bar(x2, y2, x2 + 91, y2 + 51);
206                 draw_Node(x2, y2, x2 + 90, y2 + 20, Q1);
207                 getimage(x2, y2, x2 + 90, y2 + 51, p);
208                 x = x2, y = y2, tempX = 0, tempY = 0;
209                 addX = abs(x - xRes) / 100, addY = abs(y - 150) / 100;
210                 bar(x2, y2, x2 + 91, y2 + 51);
```

```
211                 do {
212                     putimage(x + tempX, y + tempY, p, 1);
213                     delay(5);
214                     putimage(x + tempX, y + tempY, p, 1);
215                     tempX += addX;
216                     tempY += addY;
217                 } while (y + tempY < 150);
218                 putimage(x + tempX, y + tempY, p, 1);
219                 delete(p);
220                 x1 += 100;
221                 x2 += 100;
222                 xRes += 100;
223                 delay(5);
224             }
225         }
```

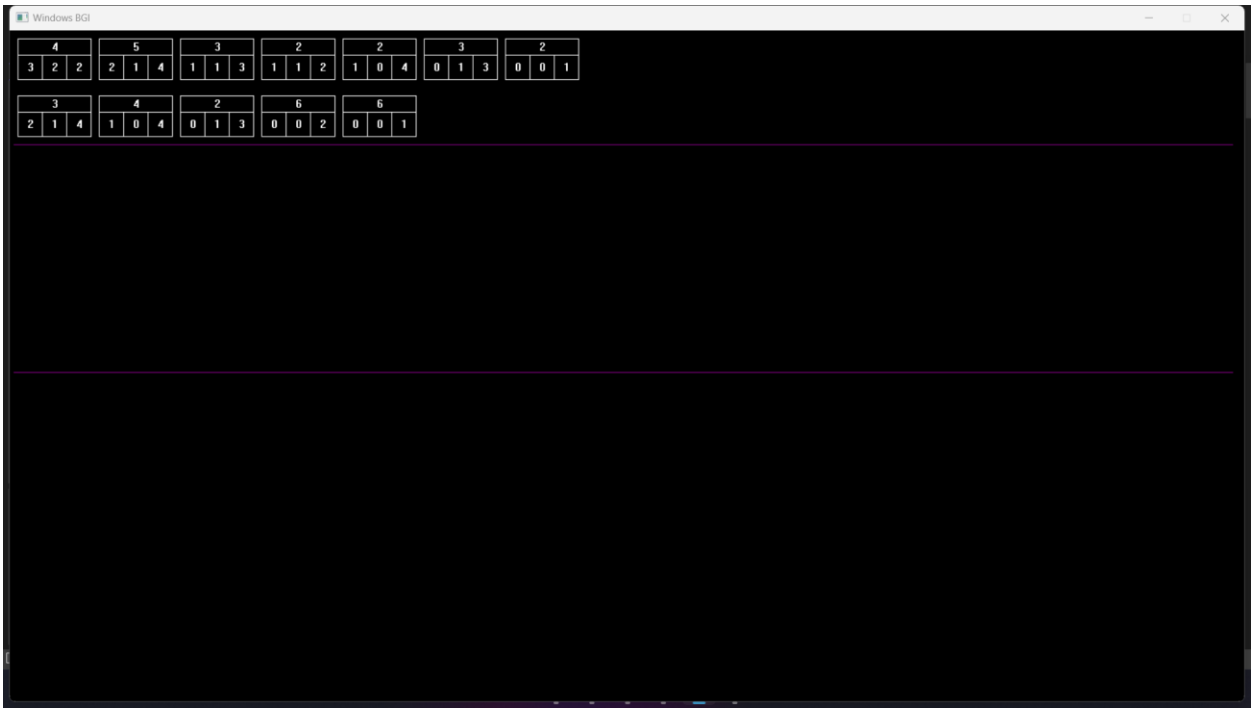
Nếu ABC(P) > ABC(Q):

```
226         else {
227             Q2->coeff = P->coeff;
228             Q2->powx = P->powx;
229             Q2->powy = P->powy;
230             Q2->powz = P->powz;
231             Q2->next = Q;
232             Q1->next = Q2;
233             Q1 = Q2;
234             create_node(Q1->coeff, Q1->powx, Q1->powy, Q1->powz, &Res);
235             P = P->next;
236             char* p;
237             int n = imagesize(0, 0, 100, 60);
238             p = (char*)malloc(n);
239             getimage(x1, y1, x1 + 90, y1 + 50, p);
240             float x = x1, y = y1, tempX = 0, tempY = 0;
241             float addX = abs(x - xRes) / 100, addY = abs(y - 150) / 100;
242             bar(x1, y1, x1 + 91, y1 + 51);
243             do {
244                 putimage(x + tempX, y + tempY, p, 1);
245                 delay(5);
246                 putimage(x + tempX, y + tempY, p, 1);
247                 tempX += addX;
248                 tempY += addY;
249             } while (y + tempY < 150);
250             putimage(x + tempX, y + tempY, p, 1);
251             delete(p);
252             x1 += 100;
253             xRes += 100;
254             delay(5);
255         }
```

g. Nhân hai đa thức

- Khởi tạo cửa sổ đồ họa, biểu diễn hai đa thức, kẻ hai đường thẳng biểu diễn phép nhân và phép cộng hai đa thức.

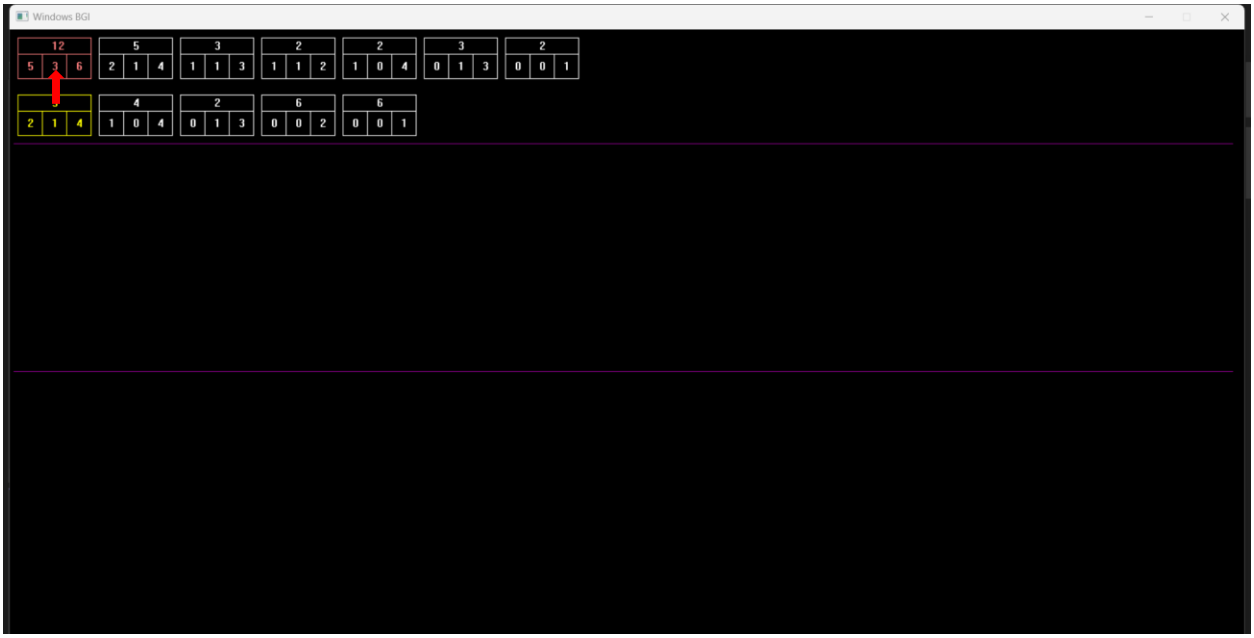
```
306     initwindow(getmaxwidth(), getmaxheight());
307     draw_Poly(10, 10, 100, 30, P);
308     delay(500);
309     draw_Poly(10, 80, 100, 100, Q);
310     delay(500);
311     setcolor(MAGENTA);
312     for (int i = 5; i <= 1405; i += 100) {
313         line(i, 140, i+100, 140);
314         delay(100);
315     }
316     for (int i = 5; i <= 1405; i += 100) {
317         line(i, 420, i + 100, 420);
318         delay(100);
319     }
```

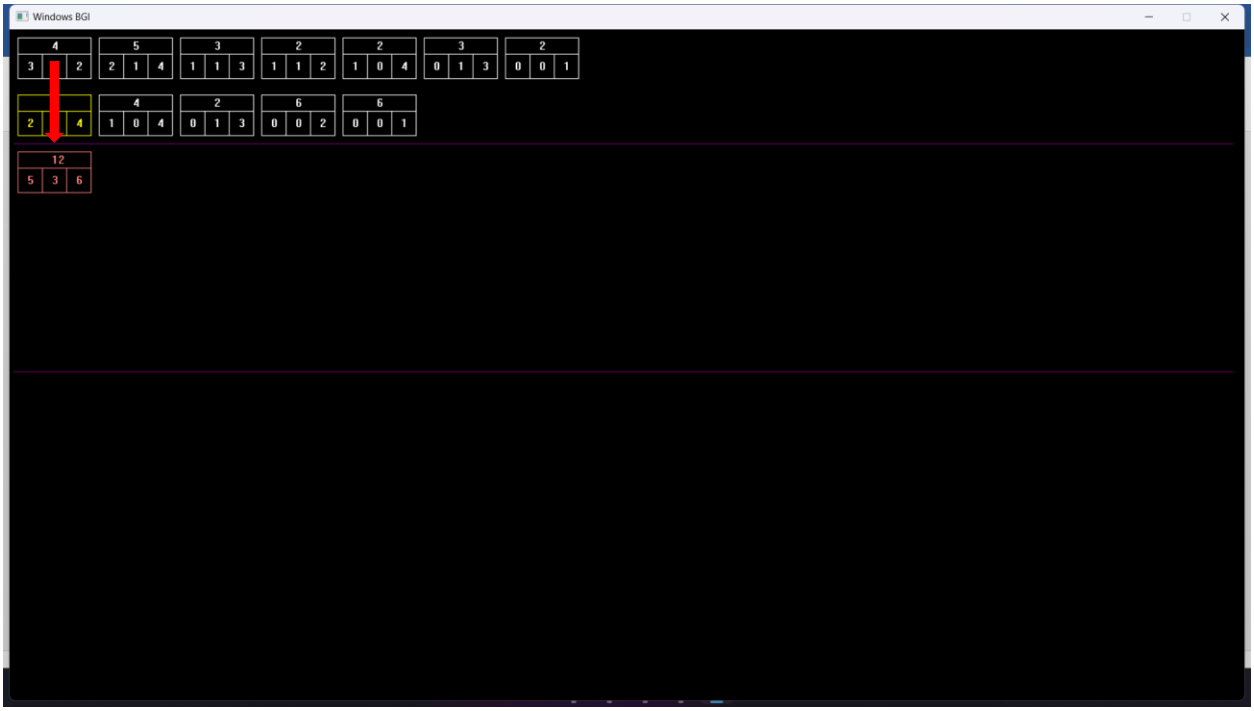


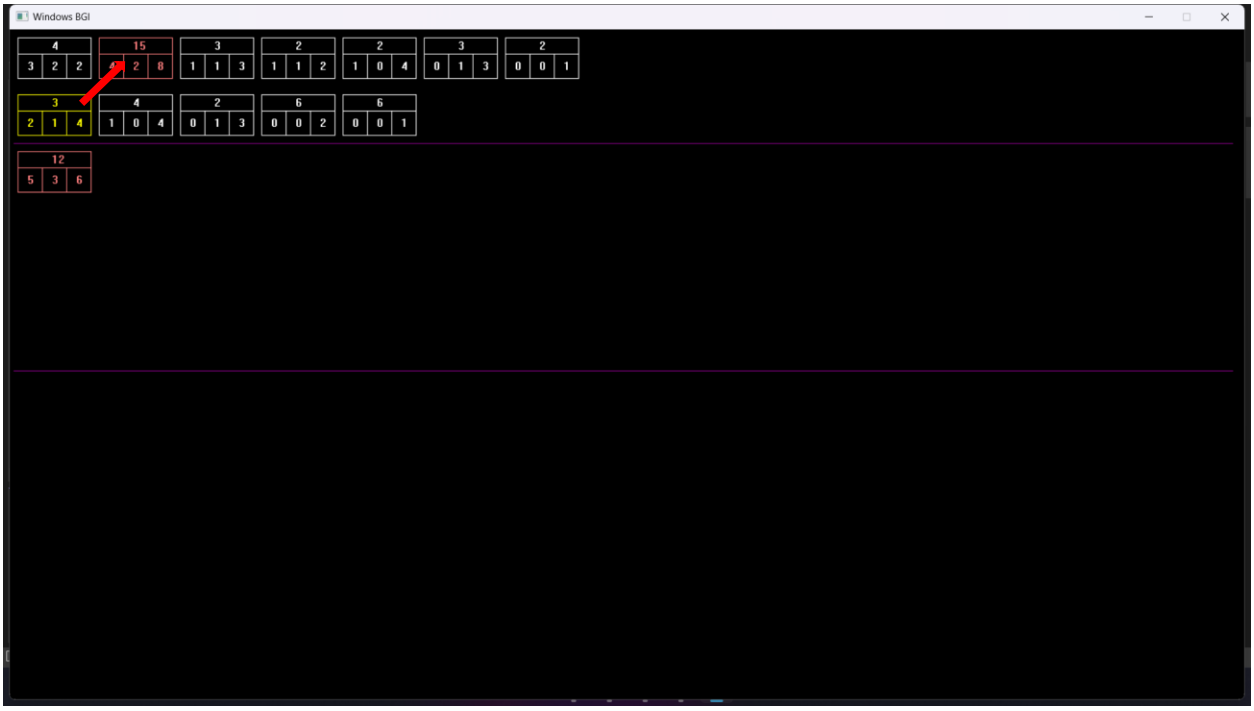
- Nhân từng hạng tử của đa thức thứ hai cho tất cả hạng tử của đa thức thứ nhất.

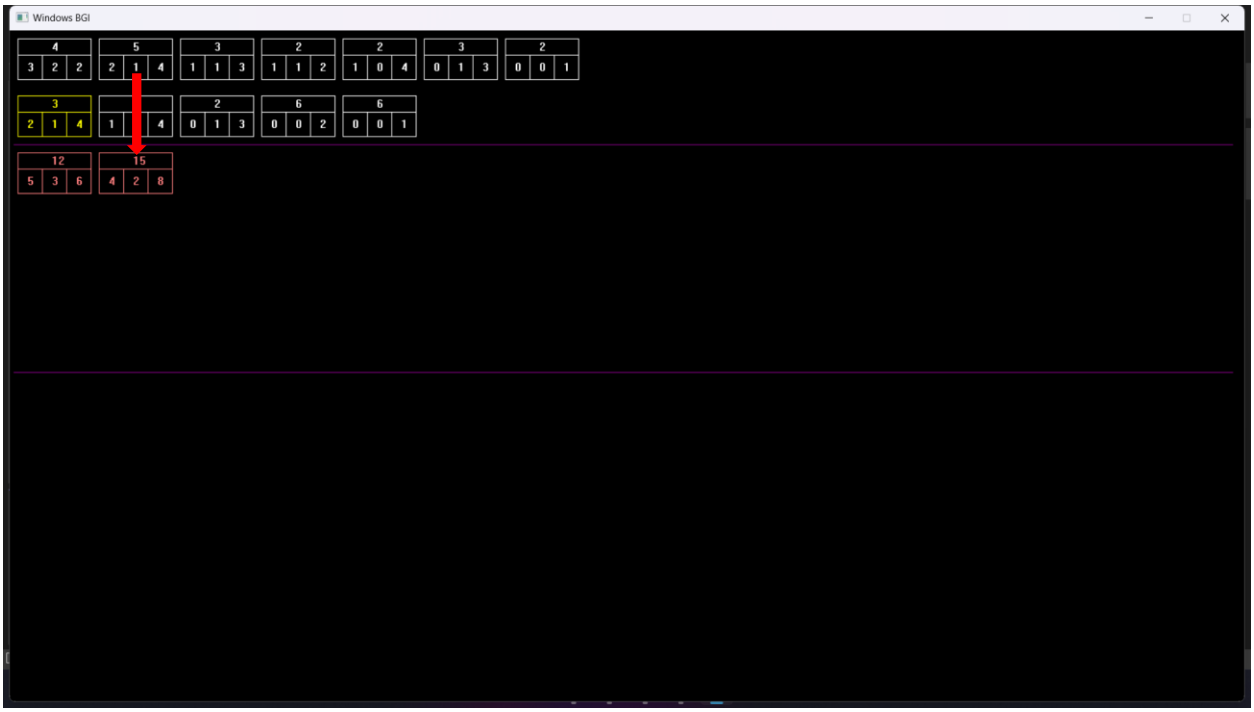
```
341     while (P->powz > 0) {
342         create_node(P->coeff * Q->coeff, P->powx + Q->powx, P->powy + Q->powy, P->powz + Q->powz, &mulNode);
343         create_node(P->coeff * Q->coeff, P->powx + Q->powx, P->powy + Q->powy, P->powz + Q->powz, &temp);
344         char* p;
345         int n = imagesize(0, 0, 100, 60);
346         p = (char*)malloc(n);
347         getimage(x2, y2, x2 + 90, y2 + 50, p);
348         float x = x2, y = y2, tempX = 0, tempY = 0;
349         float addX = abs(x - x1) / 100, addY = abs(y - y1) / 100;
350         //bar(x1, y1, x1 + 91, y1 + 51);
351         do {
352             putimage(x + tempX, y - tempY, p, 1);
353             delay(5);
354             putimage(x + tempX, y - tempY, p, 1);
355             if (x2 > x1) tempX -= addX;
356             else tempX += addX;
357             tempY += addY;
358         } while (y - tempY > y1);
359         char* p2;
360         p2 = (char*)malloc(n);
361         getimage(x1, y1, x1 + 90, y1 + 50, p2);
362         bar(x1, y1, x1 + 91, y1 + 51);
363         setcolor(LIGHTRED);
364         draw_Node(x1, y1, x1 + 90, y1 + 20, mulNode);
365         getimage(x1, y1, x1 + 90, y1 + 50, p);
366         bar(x1, y1, x1 + 91, y1 + 51);
367         putimage(x1, y1, p2, 1);
368         delete(p2);
369         x = x1, y = y1, tempX = 0, tempY = 0;
370         addX = abs(x - xMul) / 100, addY = abs(y - yMul) / 100;
```

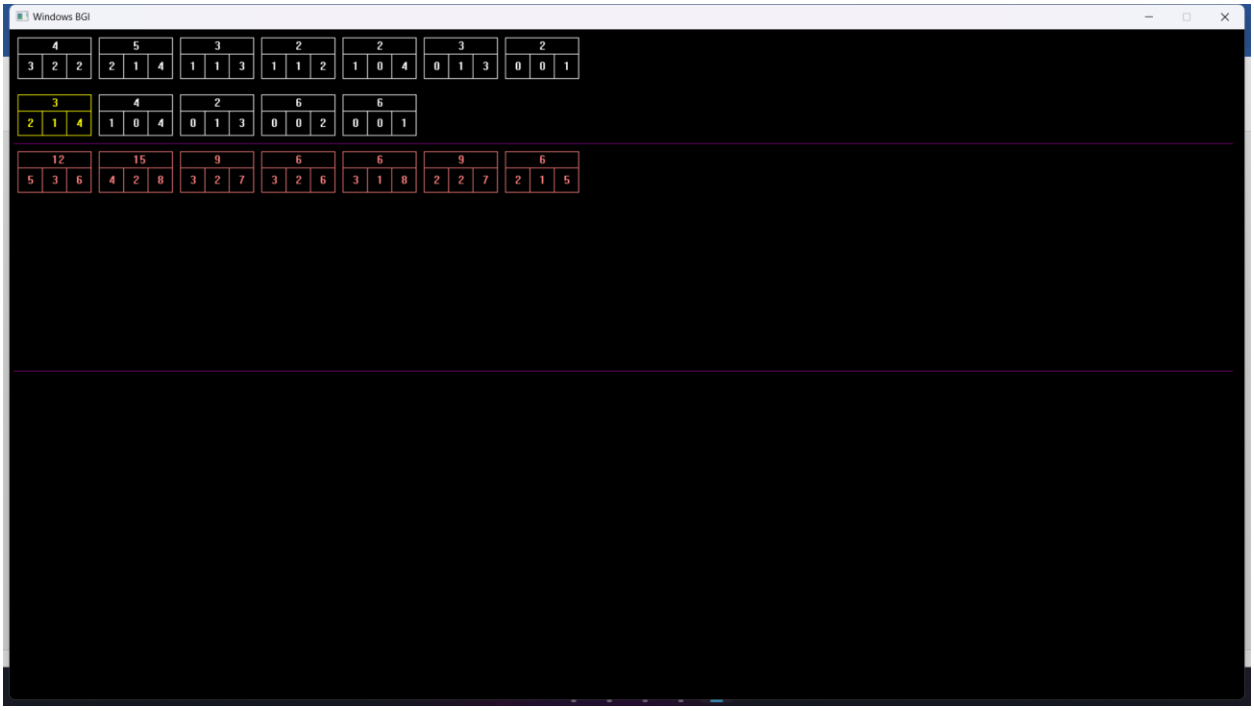
```
371             do {
372                 putimage(x + tempX, y + tempY, p, 1);
373                 delay(5);
374                 putimage(x + tempX, y + tempY, p, 1);
375                 tempX += addX;
376                 tempY += addY;
377             } while (y + tempY < yMul);
378             putimage(x + tempX, y + tempY, p, 1);
379             delete(p);
380             x1 += 100;
381             xMul += 100;
382             delay(5);
383             mulNode = nullptr;
384             p = p->next;
385         }
```






Windows BGI
4 3 2 2
15 4 2 8
3 1 1 3
2 1 1 2
2 1 0 4
3 0 1 3
2 0 0 1
3 2 1 4
4 1 0 4
2 0 1 3
6 0 0 2
6 0 0 1
12 5 3 6

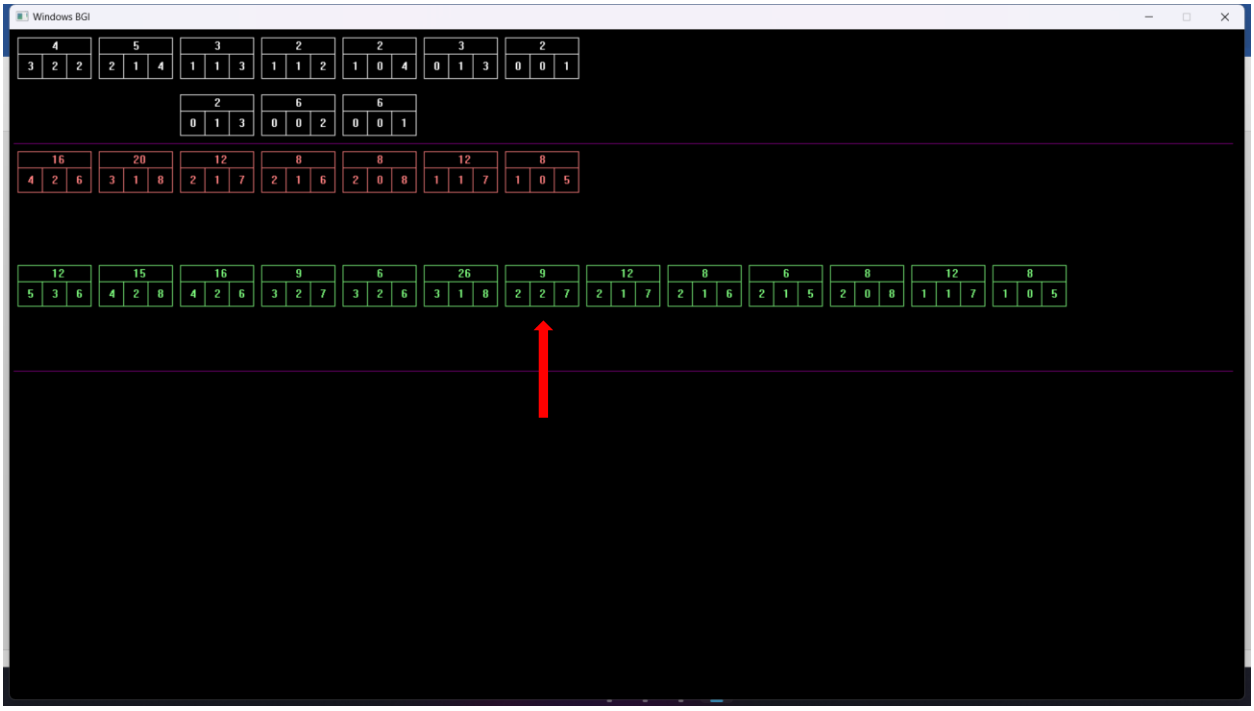

Windows BGI
4 3 2 2
5 2 1 4
3 1 1 3
2 1 1 2
2 1 0 4
3 0 1 3
2 0 0 1
3 2 1 4
2 0 1 3
6 0 0 2
6 0 0 1
12 5 3 6
15 4 2 8



- Cộng đa thức vừa nhân cho đa thức lưu trong con trỏ Res (ban đầu khởi tạo Res = 0).
- Lặp lại các bước tới khi duyệt hết hạng tử của đa thức thứ hai.


Windows BGI
4 3 2 2
5 2 1 4
3 1 1 3
2 1 1 2
2 1 0 4
3 0 1 3
2 0 0 1
4 1 0 4
2 0 1 3
6 0 0 2
6 0 0 1
12 5 3 6
15 4 2 8
9 3 2 7
6 3 2 6
6 3 1 8
9 2 2 7
6 2 1 5
12 5 3 6
15 4 2 8
9 3 2 7
6 3 2 6
6 3 1 8
9 2 2 7
6 2 1 5


Windows BGI
4 3 2 2
5 2 1 4
3 1 1 3
2 1 1 2
2 1 0 4
3 0 1 3
2 0 0 1
2 0 1 3
6 0 0 2
6 0 0 1
16 4 2 6
20 3 1 8
12 2 1 7
8 2 1 6
8 2 0 8
12 1 1 7
8 1 0 5
12 5 3 6
15 4 2 8
9 3 2 7
6 3 2 6
6 3 1 8
9 2 2 7
6 2 1 5
12 5 3 6
15 4 2 8
16 4 2 6
9 3 2 7
6 3 2 6
26 3 1 8
9 2 2 7
12 2 1 7
8 2 1 6
6 2 1 5
8 2 0 8
12 1 1 7
8 1 0 5


Windows BGI
4 3 2 2
5 2 1 4
3 1 1 3
2 1 1 2
2 1 0 4
3 0 1 3
2 0 0 1
2 0 1 3
6 0 0 2
6 0 0 1
16 4 2 6
20 3 1 8
12 2 1 7
8 2 1 6
8 2 0 8
12 1 1 7
8 1 0 5
12 5 3 6
15 4 2 8
16 4 2 6
9 3 2 7
6 3 2 6
26 3 1 8
9 2 2 7
12 2 1 7
8 2 1 6
6 2 1 5
8 2 0 8
12 1 1 7
8 1 0 5