Programming 2 - Assignment 2

Felix Dreßler (k12105003)
email FelixDressler01@gmail.com

April 26, 2022

1 Testing the Program

2 The Program - Main.cpp

```
1  #include"DistPoly.h"
2  #include<string>
3
4  string vars[2] = { "x","y" };
5
6  int main() {
7
8    // some exponent vectors("power products")
9    int e1[2] = { 1,2 }; int e2[2] = { 2,1 }; int e3[2] = { 1,0 };
10   int e4[2] = { 0,1 }; int e5[2] = { 0,0 }; int e6[2] = { 2,2 };
11
12   // construct zero polynomial in two variables, then add monomials
13   string vars[2] = { "x","y" };
14   DistPoly p(2, vars);
15   p.add(3, e1).add(5, e2).add(7, e3).add(11, e4).add(13, e5);
16
17   // construct zero polynomial in two variables, then add monomials
18   DistPoly q(2, vars);
19   q.add(11, e4).add(-3, e2).add(2, e6).add(-2, e2);
20
21   // print p and q
22   p.println();
23   q.println();
24
25   // set p to p+2*q and print it
26
27   DistPoly r = p;
28   r.add(3,e1).add(-1, e2);
29   p = r;
30   p.println();
31
32   return 0;
33 }
```

3 The Program - DistPoly.h

```
1  #pragma once
2
3  #include<string>
4
5  using namespace std;
6
7  class Monom
8  {
9  public:
10   int coeff;
11   int* exps;
12
13   //constructor
14   //Monom(int coeff, int* exps, int n);
15   //Monom(int n);
16
17   //copy constructor, copy assignment operator, destructor
18   //Monom& operator=(Monom& m);
19
20
21 };
22
23 class DistPoly
24 {
25 private:
26   int n; //number of variables
27   string* vars;
28   Monom* monoms;
29   int m; //number of potential monoms in this polynomial (allocated memory)
30   int am; //actual number of monoms in this polynomial -1
31 public:
32
33   //constructor
34   DistPoly(int n, string* vars);
35
36   //copy constructor, copy assignment operator, destructor
37   DistPoly(DistPoly& p);
38   DistPoly& operator=(DistPoly& p);
39   ~DistPoly();
40
41   DistPoly& add(int coeff, int* exps);
42   DistPoly& add(DistPoly& p);
43   //void println_brkts(); //prints the polynomial in brackets-style
44   void println();
45
46   void resize(int factor);//enlarges the polynomial,
47
48   int sort(int* exps, int n, int j);//gives back the position in which the monom with the
         exponents exps should be inserted
49 };
```

4 The Program - DistPoly.cpp

```
1  #include"DistPoly.h"
2  #include<iostream>
3
4  using namespace std;
5
6  DistPoly& DistPoly::add(int coeff, int* exps) {
7      if (coeff !=0) {
8          for (int j = 0; j <= this->m; j++) {
9
10             int k = 1;
11             if (j < this->m) {
12                 k = sort(exps, this->n, j);
13             }
14             if (k == 0) {
15                 if (this->monoms[j].coeff + coeff == 0) {
16                     for (int l = j; l < this->am+1; l++) { //shifts the monoms into the
                           empty element
17                         this->monoms[l] = this->monoms[l + 1];
18                     }
19                     this->am--;
20                 }
21                 else {
22                     this->monoms[j].coeff += coeff;
23                 }
24                 break;
25             }
26             else if (k == -1) {
27
28                 if (this->m == this->am+1) {
29                     this->resize(2);
30                 }
31
32                 for (int l = this->am + 1; l > j; l--) {
33                     this->monoms[l] = this->monoms[l-1];
34                 }
35                 this->monoms[j].coeff = coeff;
36                 this->monoms[j].exps = exps;
37                 this->am++;
38                 break;
39             }
40         }
41     }
42
43     return *this;
44 }
45
46 DistPoly& DistPoly::add(DistPoly& p) {
47
48     return *this;
49 }
50
51 int DistPoly::sort(int* exps, int n, int j) {
52     for (int i = 0; i < n; i++) {
53         if (this->monoms[j].exps[i] > exps[i]) {
54             return 1;
55         }
56          else if (this->monoms[j].exps[i] < exps[i]) {
```

```
57              return -1;
58          }
59      }
60      return 0;
61  }
62
63  void DistPoly::println() {
64      if (n == 0 || m==0 || am==0) {
65          cout << "0" << endl;
66      }
67      else {
68          for (int i = 0; i < m; i++) {
69              if (this->monoms[i].coeff != 0) {
70                  cout << this->monoms[i].coeff;
71
72                  for (int j = 0; j < n; j++) {
73                      if (this->monoms[i].exps[j] == 1) {
74                          cout << this->vars[j];
75                      }
76                      else if (this->monoms[i].exps[j] != 0) {
77                          cout << this->vars[j];
78                          cout << "^" << this->monoms[i].exps[j];
79                      }
80                  }
81                  if (i < am && this->monoms[i+1].coeff > 0) {
82                      cout << "+";
83                  }
84              }
85          }
86          cout << "\n";
87      }
88  }
89
90  DistPoly::DistPoly(int n, string* vars) {
91      this->n = n;
92      this->vars = vars;
93      this->m = 1;
94      this->am = 0;
95      this->monoms = new Monom[m];
96      for (int j = 0; j < m; j++) {
97          this->monoms[j].coeff = 0;
98          this->monoms[j].exps = new int[n];
99          for (int i = 0; i < n; i++) {
100             this->monoms[j].exps[i] = 0;
101         }
102     }
103 }
104
105 DistPoly::DistPoly(DistPoly& p) {
106     this->n = p.n;
107     this->vars = p.vars;
108     this->m = p.m;
109     this->am = p.am;
110     this->monoms = new Monom[(p.m) + 1];
111     for (int i = 0; i < m; i++) {
112         this->monoms[i].coeff = p.monoms[i].coeff;
113         this->monoms[i].exps = p.monoms[i].exps;
114     }
115
116 }
```

```
117
118 DistPoly& DistPoly::operator=(DistPoly& p) {
119     this->n = p.n;
120     this->vars = p.vars;
121     this->m = p.m;
122     this->am = p.am;
123     this->monoms = new Monom[(p.m) + 1];
124     for (int i = 0; i < m; i++) {
125         this->monoms[i].coeff = p.monoms[i].coeff;
126         this->monoms[i].exps = p.monoms[i].exps;
127     }
128     return *this;
129 }
130
131 //problem is because it tries to delete a pointer, that has already been deleted ->
        better: create constructor and destructor for class Monom
132 DistPoly::~DistPoly() {
133     //for (int i = 0; i < this->m; i++) {
134     //    delete[] this->monoms[i].exps;
135     //}
136     delete[] this->monoms;
137 }
138
139 void DistPoly::resize(int factor) {
140     Monom* NewMonoms = new Monom[factor * m];
141     for (int i = 0; i < this->m; i++) {
142         NewMonoms[i] = this->monoms[i];
143     }
144     for (int i = this->m; i < (this->m) * factor; i++) {//initializes the remaining
            elements of the array with the standard value 0
145         NewMonoms[i].coeff = 0;
146         NewMonoms[i].exps = new int[this->n];
147         for (int j = 0; j < this->n; j++) {
148             NewMonoms[i].exps[j] = 0;
149         }
150     }
151     delete[] this->monoms;
152     this->monoms = NewMonoms;
153     this->m = factor * (this->m);
154 }
155 //idea: create new array of monoms with emty constructor, then initialize them by
        assigning a newly constructed monom (with different constructor) to every element of
        the array
156
157
158 //constructor
159 //Monom::Monom(int coeff, int* exps, int n) {
160 //    this->coeff = coeff;
161 //    this->exps = new int[n]; //creates a new array of exponents, this is in order to
        have seperate pointers and deallocate their respectivve memory later (for DistPoly)
162 //    this->exps = exps;
163 //}
164 //
165 //
166 ////constructor
167 //Monom::Monom() {
168 //    this->coeff = 0;
169 //    this->exps = 0;
170 //}
171 //
```

```
172 //
173 ////copy assignment operator
174 //Monom& Monom::operator=(Monom& m) {
175 //    this->coeff = m.coeff;
176 //    this->exps = m.exps;
177 //
178 //    return *this;
179 //}
```