

LAPORAN PRAKTIKUM

“Pertemuan ke-5: POST TEST – Variasi Link List”

Diajukan untuk memenuhi salah satu praktikum Mata Struktur Data Informatika yang di
ampu oleh:

Dr., Ardiansyah, S.T., M.Cs.



Disusun Oleh:

Mohammad Farid Hendianto 2200018401

A / Rabu 10.30 – 13.30 Lab. Jaringan

**PROGRAM STUDI INFORMATIKA
UNIVERSITAS AHMAD DAHLAN
FAKULTAS TEKNOLOGI INDUSTRI
TAHUN 2023**

Ubah konstruktor dan operasi dari link list posttest pertemuan 4 dengan model link list berikut:

a. no ganjil : double link list

Berikut adalah sourcecode kodingan:

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void errorInput() {
5     cout << "Input tidak valid" << endl;
6 }
7
8 struct Data {
9     string province;
10    string city;
11    string nik;
12    string name;
13    struct Bore {
14        string place;
15        int date;
16        int month;
17        int year;
18    } bore;
19
20    bool gender;
21    string bloodType;
22
23    struct Address {
24        string address;
25        int neighborhoodAssociation;
26        int communityAssociation;
27        string district;
28        string subdistrict;
29    } address;
30    string religion;
31    string isKorid;
32    string jom;
33    string citizenship;
34    string validMtl;
35    struct Kristen {
36        int date;
37        int month;
38        int year;
39    } kristen;
40    int created;
41 };
42
43 struct Node {
44    Data data;
45    Node* next;
46    Node* prev;
47 };
48
49 class KLP {
50 private:
51     Node* head = nullptr;
52     Node* tail = nullptr;
53     int count = 0;
54 public:
55
56     bool validateProvince(string province) {
57         return province.length() <= 50;
58     }
59
60     bool validateCity(string city) {
61         return city.length() <= 50;
62     }
63
64     bool validateNik(string nik) {
65         return nik.length() <= 16;
66     }
67
68     bool validateName(string name) {
69         return name.length() <= 50;
70     }
71
72     bool validateBorePlace(string borePlace) {
73         return borePlace.length() <= 20;
74     }
75
76     bool validateDate(int date) {
77         return date >= 1 && date <= 31;
78     }
79
80     bool validateMonth(int month) {
81         return month >= 1 && month <= 12;
82     }
83
84     bool validateYear(int year) {
85         return year >= 1900 && year <= 2023;
86     }
87
88     bool validateGender(int gender) {
89         return gender == 0 || gender == 1;
90     }
91
92     bool validateBloodType(string bloodType) {
93         return bloodType == "A" || bloodType == "B" || bloodType == "AB" || bloodType == "O";
94     }
95
96     bool validateAddress(string address) {
97         return address.length() <= 100;
98     }
99
100    bool validateNeighborhoodAssociation(int neighborhoodAssociation) {
101        return neighborhoodAssociation >= 0 && neighborhoodAssociation <= 999;
102    }
103
104    bool validateCommunityAssociation(int communityAssociation) {
105        return communityAssociation >= 0 && communityAssociation <= 999;
106    }
107
108    bool validateSubdistrict(string subdistrict) {
109        return subdistrict.length() <= 20;
110    }
111
112    bool validateDistrict(string district) {
113        return district.length() <= 20;
114    }
115
116    bool validateReligion(string religion) {
117        return religion == "Islam" || religion == "Kristen" || religion == "Katolik" || religion == "Hindu" || religion == "Budha" || religion == "Khonghucu";
118    }
119
120    bool validateKorid(string isKorid) {
121        return isKorid == "Korid" || isKorid == "Belum Korid";
122    }
123
124    bool validateJom(string jom) {
125        return jom.length() <= 20;
126    }
127
128    bool validateCitizenship(string citizenship) {
129        return citizenship.length() <= 20;
130    }
131
132    bool validateValidMtl(string validMtl) {
133        return validMtl.length() <= 20;
134    }
135
136    bool validateKristen(string kristen) {
137        return kristen.date >= 1 && kristen.date <= 31 && kristen.month >= 1 && kristen.month <= 12 && kristen.year >= 1900 && kristen.year <= 2023;
138    }
139
140    bool validateCreated(int created) {
141        return created >= 0 && created <= 1000000;
142    }
143
144    void addNode(Data data) {
145        Node* newNode = new Node;
146        newNode->data = data;
147        if (head == nullptr) {
148            head = newNode;
149            tail = newNode;
150        } else {
151            tail->next = newNode;
152            newNode->prev = tail;
153            tail = newNode;
154        }
155        count++;
156    }
157
158    void display() {
159        Node* temp = head;
160        while (temp != nullptr) {
161            cout << "Data: " << temp->data << endl;
162            temp = temp->next;
163        }
164    }
165
166    void deleteNode(int index) {
167        if (index < 0 || index >= count) {
168            cout << "Index tidak valid" << endl;
169            return;
170        }
171        if (index == 0) {
172            head = head->next;
173        } else if (index == count - 1) {
174            tail = tail->prev;
175        } else {
176            Node* temp = head;
177            for (int i = 0; i < index; i++) {
178                temp = temp->next;
179            }
180            temp->prev->next = temp->next;
181            temp->next->prev = temp->prev;
182            delete temp;
183        }
184        count--;
185    }
186
187    void search(string keyword) {
188        Node* temp = head;
189        while (temp != nullptr) {
190            if (temp->data.province == keyword) {
191                cout << "Province: " << keyword << endl;
192            }
193            temp = temp->next;
194        }
195    }
196
197    void sort() {
198        Node* temp = head;
199        Node* sortedHead = nullptr;
200        while (temp != nullptr) {
201            Node* current = temp;
202            temp = temp->next;
203            current->next = sortedHead;
204            sortedHead = current;
205        }
206        head = sortedHead;
207    }
208
209    void reverse() {
210        Node* temp = head;
211        Node* prev = nullptr;
212        while (temp != nullptr) {
213            Node* next = temp->next;
214            temp->next = prev;
215            prev = temp;
216            temp = next;
217        }
218        head = prev;
219    }
220
221    void clear() {
222        head = nullptr;
223        tail = nullptr;
224        count = 0;
225    }
226
227    int getCount() {
228        return count;
229    }
230
231    void printCount() {
232        cout << "Total Data: " << count << endl;
233    }
234
235    ~KLP() {
236        clear();
237    }
238 };
239
240 int main() {
241     KLP klp;
242     Data data;
243     data.province = "Jawa Barat";
244     data.city = "Bandung";
245     data.nik = "3170010010000000";
246     data.name = "John Doe";
247     data.bore.place = "Bandung";
248     data.bore.date = 1;
249     data.bore.month = 1;
250     data.bore.year = 2023;
251     data.gender = 1;
252     data.bloodType = "A";
253     data.address.address = "Jl. Merdeka No. 123";
254     data.address.neighborhoodAssociation = 123;
255     data.address.communityAssociation = 456;
256     data.address.district = "Kota Bandung";
257     data.address.subdistrict = "Kecamatan Bandung";
258     data.religion = "Islam";
259     data.isKorid = "Korid";
260     data.jom = "Jember";
261     data.citizenship = "Kewarganegaraan Indonesia";
262     data.validMtl = "Valid";
263     data.kristen.date = 1;
264     data.kristen.month = 1;
265     data.kristen.year = 2023;
266     data.created = 1000000;
267     klp.addNode(data);
268     klp.display();
269     klp.deleteNode(0);
270     klp.search("Jawa Barat");
271     klp.sort();
272     klp.reverse();
273     klp.clear();
274     klp.printCount();
275     return 0;
276 }

```

```

127 void input() {
128     Node* newnode = new Node;
129     do {
130         cout << "Province: ";
131         getline(cin, newnode->data.province);
132         if (!validateProvince(newnode->data.province)) {
133             errorInput();
134         }
135     } while (!validateProvince(newnode->data.province));
136     do {
137         cout << "Kota: ";
138         getline(cin, newnode->data.city);
139         if (!validateCity(newnode->data.city)) {
140             errorInput();
141         }
142     } while (!validateCity(newnode->data.city));
143     do {
144         cout << "NIK: ";
145         getline(cin, newnode->data.nik);
146         if (!validateNik(newnode->data.nik)) {
147             errorInput();
148         }
149     } while (!validateNik(newnode->data.nik));
150     do {
151         cout << "Nama: ";
152         getline(cin, newnode->data.name);
153         if (!validateName(newnode->data.name)) {
154             errorInput();
155         }
156     } while (!validateName(newnode->data.name));
157     do {
158         cout << "tempat lahir: ";
159         getline(cin, newnode->data.born_place);
160         if (!validateBornPlace(newnode->data.born_place)) {
161             errorInput();
162         }
163     } while (!validateBornPlace(newnode->data.born_place));
164     do {
165         cout << "tanggal lahir: ";
166         cin >> newnode->data.born_date;
167         if (!validateDate(newnode->data.born_date)) {
168             errorInput();
169         }
170     } while (!validateDate(newnode->data.born_date));
171     do {
172         cout << "bulan lahir: ";
173         cin >> newnode->data.born_month;
174         if (!validateMonth(newnode->data.born_month)) {
175             errorInput();
176         }
177     } while (!validateMonth(newnode->data.born_month));
178     do {
179         cout << "tahun lahir: ";
180         cin >> newnode->data.born_year;
181         if (!validateYear(newnode->data.born_year)) {
182             errorInput();
183         }
184     } while (!validateYear(newnode->data.born_year));
185     do {
186         cout << "jenis kelamin (laki-laki, perempuan): ";
187         cin >> newnode->data.gender;
188         if (!validateGender(newnode->data.gender)) {
189             errorInput();
190         }
191     } while (!validateGender(newnode->data.gender));
192     cin.ignore();
193     do {
194         cout << "tipe darah: ";
195         getline(cin, newnode->data.bloodtype);
196         if (!validateBloodType(newnode->data.bloodtype)) {
197             errorInput();
198         }
199     } while (!validateBloodType(newnode->data.bloodtype));
200     do {
201         cout << "alamat: ";
202         getline(cin, newnode->data.address.address);
203         if (!validateAddress(newnode->data.address.address)) {
204             errorInput();
205         }
206     } while (!validateAddress(newnode->data.address.address));
207     do {
208         cout << "RT: ";
209         cin >> newnode->data.address.neighborhoodAssociation;
210         if (!validateNeighborhoodAssociation(newnode->data.address.neighborhoodAssociation)) {
211             errorInput();
212         }
213     } while (!validateNeighborhoodAssociation(newnode->data.address.neighborhoodAssociation));
214     do {
215         cout << "RW: ";
216         cin >> newnode->data.address.communityAssociation;
217         if (!validateCommunityAssociation(newnode->data.address.communityAssociation)) {
218             errorInput();
219         }
220     } while (!validateCommunityAssociation(newnode->data.address.communityAssociation));
221     cin.ignore();
222     do {
223         cout << "kel/Desa: ";
224         getline(cin, newnode->data.address.subDistrict);
225         if (!validateSubDistrict(newnode->data.address.subDistrict)) {
226             errorInput();
227         }
228     } while (!validateSubDistrict(newnode->data.address.subDistrict));
229     do {
230         cout << "kecamatan: ";
231         getline(cin, newnode->data.address.district);
232         if (!validateDistrict(newnode->data.address.district)) {
233             errorInput();
234         }
235     } while (!validateDistrict(newnode->data.address.district));
236     do {
237         cout << "Agama (Islam, Kristen, Katolik, Hindu, Buddha, Konghucu): ";
238         getline(cin, newnode->data.religion);
239         if (!validateReligion(newnode->data.religion)) {
240             errorInput();
241         }
242     } while (!validateReligion(newnode->data.religion));
243     do {
244         cout << "status Perkawinan (kawin, belum kawin): ";
245         getline(cin, newnode->data.married);
246         if (!validateMarried(newnode->data.married)) {
247             errorInput();
248         }
249     } while (!validateMarried(newnode->data.married));
250     do {
251         cout << "pekerjaan: ";
252         getline(cin, newnode->data.job);
253         if (!validateJob(newnode->data.job)) {
254             errorInput();
255         }
256     } while (!validateJob(newnode->data.job));
257     newnode->data.citizenship = "Indo";
258     newnode->data.validUntil = "SEKELANG HAYAT";
259     time_t now = time(0);
260     tm *ltm = localtime(&now);
261     newnode->data.created_date = ltm->tm_mday;
262     newnode->data.created_month = ltm->tm_mon + 1;
263     newnode->data.created_year = ltm->tm_year + 1900;
264     for (int i = 0; i < newnode->data.religion.length(); i++) {
265         newnode->data.religion[i] = toupper(newnode->data.religion[i]);
266     }
267     for (int i = 0; i < newnode->data.married.length(); i++) {
268         newnode->data.married[i] = toupper(newnode->data.married[i]);
269     }
270 }

```

```

272 ofstream file;
273 file.open("data.txt", ios::app);
274 file << "Provinsi ";
275 file << "Kabupaten ";
276 file << "NIK ";
277 file << "Nama ";
278 file << "Tempat/Tgl lahir ";
279 file << "Jenis Kelamin ";
280 file << "Gol Darah ";
281 file << "Alamat ";
282 file << "RT/RW ";
283 file << "KelDesa ";
284 file << "Kecamatan ";
285 file << "Agama ";
286 file << "Status Perkawinan ";
287 file << "Pekerjaan ";
288 file << "Kewarganegaraan ";
289 file << "SuratLain Higgs ";
290 file << "TglBuat ";
291 file << "-----" << endl;
292 file.close();
293 Node* head = nullptr;
294 Node* tail = nullptr;
295
296 newnode->next = nullptr;
297 if (head == nullptr) {
298     head = newnode;
299     tail = newnode;
300     newnode->prev = nullptr;
301 } else {
302     tail->next = newnode;
303     newnode->prev = tail;
304     tail = newnode;
305 }
306 count++;
307
308 void output() {
309     cout << "-----" << endl;
310     cout << "KARTU TANPA PENCORAN" << endl;
311     cout << "-----" << endl;
312     Node* current = head;
313     while (current != nullptr) {
314         int provinceLength = current->data.province.length();
315         int cityLength = current->data.city.length();
316         int provinceLength + cityLength + provinceLength + cityLength;
317         for (int i = 0; i < (provinceLength + cityLength) / 2; i++) {
318             cout << " ";
319         }
320         cout << " " << current->data.province << endl;
321         for (int j = 0; j < (cityLength + cityLength) / 2; j++) {
322             cout << " ";
323         }
324         cout << " " << current->data.city << endl;
325         cout << "NIK " << current->data.nik << endl;
326         cout << "Nama " << current->data.name << endl;
327         cout << "Tempat/Tgl lahir " << current->data.bornPlace << " " << setfill('0') << setw(2) << current->data.born.date << " " << setfill('0') << setw(2) << current->data.born.month << " " << setfill('0') << setw(2) << current->data.born.year << endl;
328         cout << setfill(' ') << setw(35) << current->data.bloodType << " " << "Gol Darah " << current->data.bloodType << endl;
329         cout << "Alamat " << current->data.address.address << endl;
330         cout << "RT/RW " << setfill('0') << setw(3) << current->data.address.neighborhoodAssociation << " " << setfill('0') << setw(3) << current->data.address.communityAssociation << endl;
331         cout << "KelDesa " << current->data.address.subDistrict << endl;
332         cout << "Kecamatan " << current->data.address.district << endl;
333         cout << "Agama " << current->data.religion << endl;
334         cout << "Status Perkawinan " << current->data.married << endl;
335         cout << setfill(' ') << setw(47) << "Pekerjaan " << endl;
336         cout << "Pekerjaan " << current->data.job << endl;
337         cout << "Kewarganegaraan " << current->data.citizenship << endl;
338         cout << setfill(' ') << setw(48) << "SuratLain Higgs (current->data.created.date).length() << " " << setfill('0') << setw(2) << current->data.created.date << " " << setfill('0') << setw(2) << current->data.created.month << " " << setfill('0') << setw(2) << current->data.created.year << endl;
339         cout << setfill(' ') << setw(40) << "TglBuat " << endl;
340         cout << "-----" << endl;
341         current = current->next;
342     }
343 }
344
345 void initialize() {
346     ifstream file;
347     file.open("data.txt");
348     string line;
349     count = 0;
350     while (getline(file, line)) {
351         if (line.find("NIK") != string::npos) {
352             count++;
353         }
354     }
355     cout << "SuratLain Higgs " << count << endl;
356     file.close();
357     this->head = nullptr;
358     Node* tail = nullptr;
359     for (int i = 0; i < count; i++) {
360         Data newdata;
361         while (getline(file, line)) {
362             if (line.find("Provinsi") != string::npos) newdata.province = line.substr(line.find(" ") + 2);
363             else if (line.find("Kabupaten") != string::npos) newdata.city = line.substr(line.find(" ") + 2);
364             else if (line.find("NIK") != string::npos) newdata.nik = line.substr(line.find(" ") + 2);
365             else if (line.find("Nama") != string::npos) newdata.name = line.substr(line.find(" ") + 2);
366             else if (line.find("Tempat/Tgl lahir") != string::npos) {
367                 newdata.bornPlace = line.substr(line.find(" ") + 2, line.find(" ") - line.find(" ") - 2);
368                 newdata.born.date = stoi(line.substr(line.find(" ") + 2, 2));
369                 newdata.born.month = stoi(line.substr(line.find(" ") + 2, 2));
370                 newdata.born.year = stoi(line.substr(line.find(" ") + 2, 2));
371             }
372             else if (line.find("Gol Darah") != string::npos) {
373                 newdata.bloodType = line.substr(line.find(" ") + 2);
374             }
375             else if (line.find("Alamat") != string::npos) newdata.address.address = line.substr(line.find(" ") + 2);
376             else if (line.find("RT/RW") != string::npos) newdata.address.neighborhoodAssociation = stoi(line.substr(line.find(" ") + 2, 3));
377             else if (line.find("KelDesa") != string::npos) newdata.address.subDistrict = line.substr(line.find(" ") + 2);
378             else if (line.find("Kecamatan") != string::npos) newdata.address.district = line.substr(line.find(" ") + 2);
379             else if (line.find("Agama") != string::npos) newdata.religion = line.substr(line.find(" ") + 2);
380             else if (line.find("Status Perkawinan") != string::npos) newdata.married = line.substr(line.find(" ") + 2);
381             else if (line.find("Pekerjaan") != string::npos) newdata.job = line.substr(line.find(" ") + 2);
382             else if (line.find("Kewarganegaraan") != string::npos) newdata.citizenship = line.substr(line.find(" ") + 2);
383             else if (line.find("SuratLain Higgs") != string::npos) newdata.validUntil = line.substr(line.find(" ") + 2);
384             else if (line.find("TglBuat") != string::npos) {
385                 newdata.created.date = stoi(line.substr(line.find(" ") + 2, 2));
386                 newdata.created.month = stoi(line.substr(line.find(" ") + 2, 2));
387                 newdata.created.year = stoi(line.substr(line.find(" ") + 2, 2));
388             }
389             else if (line.find("-----") != string::npos) {
390                 Node* newnode = new Node(newdata, nullptr, tail);
391                 if (head == nullptr) {
392                     head = newnode;
393                     tail = newnode;
394                 }
395                 else {
396                     tail->next = newnode;
397                     newnode->prev = tail;
398                     tail = newnode;
399                 }
400                 break;
401             }
402         }
403     }
404 }
405
406 file.close();
407 }

```

```

422 void delAtHead()
423 {
424     if(head == nullptr){
425         cout << "Data kosong" << endl;
426         return;
427     }
428     Node* temp = head;
429     head = head->next;
430     if(head != nullptr){
431         head->prev = nullptr;
432     }
433     delete temp;
434     cout << "Data berhasil dihapus" << endl;
435     count--;
436 }
437
438 ifstream file;
439 file.open("data.txt");
440 ofstream tempFile;
441 tempFile.open("temp.txt");
442 string line;
443 int lineCount = 0;
444 while (getline(file, line)) {
445     if (line.find("link") != string::npos) {
446         lineCount++;
447     }
448     if (lineCount != 1) {
449         tempFile << line << endl;
450     }
451 }
452 file.close();
453 tempFile.close();
454 remove("data.txt");
455 rename("temp.txt", "data.txt");
456 }
457
458 void delAtTail()
459 {
460     if(head == nullptr){
461         cout << "linked list kosong" << endl;
462         return;
463     }
464     Node* current = head;
465     while(current->next != nullptr){
466         current = current->next;
467     }
468     if(current->prev == nullptr){
469         head = nullptr;
470     } else {
471         current->prev->next = nullptr;
472     }
473     delete current;
474     cout << "Data berhasil dihapus" << endl;
475     count--;
476 }
477
478 ifstream file;
479 file.open("data.txt");
480 ofstream tempFile;
481 tempFile.open("temp.txt");
482 string line;
483 int lineCount = 0;
484 while (getline(file, line)) {
485     if (line.find("link") != string::npos) {
486         lineCount++;
487     }
488     if (lineCount != count + 1) {
489         tempFile << line << endl;
490     }
491 }
492 file.close();
493 tempFile.close();
494 remove("data.txt");
495 rename("temp.txt", "data.txt");
496 }
497 }
498
499 int main() {
500     KLP klp;
501     int choice;
502     while(true){
503         klp.initialise();
504         cout << "Menu: " << endl;
505         cout << "1. Input Data" << endl;
506         cout << "2. Tampilkan Data" << endl;
507         cout << "3. Delete Data" << endl;
508         cout << "4. Keluar" << endl;
509         cout << "Pilih: ";
510         cin >> choice;
511         system("cls");
512         cin.ignore();
513         switch (choice) {
514             case 1:
515                 klp.input();
516                 cout << "Data berhasil diinput" << endl;
517                 break;
518             case 2:
519                 klp.output();
520                 break;
521             case 3:
522                 while(true){
523                     system("cls");
524                     cout << "Menu: " << endl;
525                     cout << "1. Delete data di depan" << endl;
526                     cout << "2. Delete data di belakang" << endl;
527                     cout << "3. Kembali" << endl;
528                     cout << "Pilih: ";
529                     cin >> choice;
530                     switch(choice){
531                         case 1:
532                             klp.delAtHead();
533                             break;
534                         case 2:
535                             klp.delAtTail();
536                             break;
537                         case 3:
538                             break;
539                         default:
540                             cout << "Pilihan tidak valid" << endl;
541                             system("pause");
542                             break;
543                     }
544                 }
545                 break;
546             case 4:
547                 return EXIT_SUCCESS;
548             default:
549                 cout << "Pilihan tidak valid" << endl;
550                 break;
551         }
552         system("pause");
553         system("cls");
554     }
555 }

```

The screenshot shows a code editor with the following C++ code:

```

#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int sum = 0;
    for (int i = 1; i <= n; i++)
    {
        sum += i*i;
    }
    cout << sum << endl;
    return 0;
}


```

The output window shows the execution results:

```

10
385

```



```
Banyak KTP: 10
Menu:
1. Input Data
2. Tampilkan Data
3. Delete Data
4. Keluar
Pilihan: █
```

Menu KTP

Menu KTP

```

Agama (Islam, Kristen, Katolik, Hindu, Buddha, Konghucu): Islam
Provinsi: PROVINSI BANTEN
Kota: KABUPATEN PANDEGLANG
NIK: 3601120705880002
Nama: RUSLI
Tempat Lahir: PANDEGLANG
Tanggal Lahir: 17
Bulan Lahir: 7
Tahun Lahir: 1988
Jenis kelamin (0=laki-laki, 1=perempuan): 0
Golongan Darah: A
Alamat: KP BADONGAN
RT: 1
RW: 13
Kel/Desa: TELUK
Kecamatan: LABUAN
Agama (Islam, Kristen, Katolik, Hindu, Buddha, Konghucu): Islam
Status Perkawinan (Kawin, Belum Kawin): Belum Kawin
Pekerjaan: WIRASWASTA
Data berhasil diinput
Press any key to continue . . . █
    
```

Menambahkan KTP baru dengan Doubly Linked List



```

Kewarganegaraan : WNI
Berlaku Hingga : SEUMUR HIDUP
17-10-2023
TTD
=====
PROVINSI JAWA TENGAH
KOTA SURAKARTA
NIK : 3372011210770001
Nama : SELAMAT IDUL ADHA
Tempat/Tgl Lahir : SURAKARTA, 12-10-1977
Jenis Kelamin : PEREMPUAN Gol.Darah : AB
Alamat : BARON CILIK
RT/RW : 004/005
Kel/Desa : BUMI
Kecamatan : LAWEYAN
Agama : ISLAM
Status Perkawinan : KAWIN Foto.jpg
Pekerjaan : PERDAGANGAN
Kewarganegaraan : WNI
Berlaku Hingga : SEUMUR HIDUP
17-10-2023
TTD
=====
PROVINSI BANTEN
KABUPATEN PANDEGLANG
NIK : 3601120705880002
Nama : RUSLI
Tempat/Tgl Lahir : PANDEGLANG, 17-07-1988
Jenis Kelamin : PEREMPUAN Gol.Darah : A
Alamat : KP BADONGAN
RT/RW : 001/013
Kel/Desa : TELUK
Kecamatan : LABUAN
Agama : ISLAM
Status Perkawinan : BELUM KAWIN Foto.jpg
Pekerjaan : WIRASWASTA
Kewarganegaraan : WNI
Berlaku Hingga : SEUMUR HIDUP
31-10-2023
TTD
=====
Press any key to continue . . .

```

KTP setelah ditambahkan

```

Menu:
1. Delete data di depan
2. Delete data di belakang
3. Kembali
Pilihan:

```

Mencoba menu 2. Delete

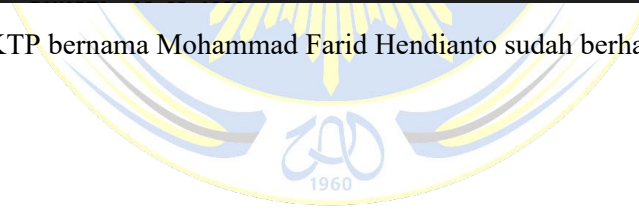
KARTU TANDA PENDUDUK			
=====			
PROVINSI KALIMANTAN TENGAH			
KABUPATEN KOTAWARINGIN BARAT			
NIK	: 3674072703040003		
Nama	: MOHAMMAD FARID HENDIANTO		
Tempat/Tgl Lahir	: TANGERANG, 27-03-2004		
Jenis Kelamin	: PEREMPUAN	Gol.Darah : B	
Alamat	: BATAN INDAH BLOK J-14		
RT/RW	: 003/004		
Kel/Desa	: KADEMANGAN		
Kecamatan	: SETU		
Agama	: Islam		
Status Perkawinan	: Belum Kawin	Foto.jpg	
Pekerjaan	: PELAJAR/MAHASISWA		
Kewarganegaraan	: WNI	27-05-2021	
Berlaku Hingga	: SEUMUR HIDUP	TTD	
=====			
PROVINSI DAERAH ISTIMEWA YOGYAKARTA			
KOTA YOGYAKARTA			
NIK	: 3471135407040001		
Nama	: AISYAH SYAFI'I NURJANNAH		
Tempat/Tgl Lahir	: YOGYAKARTA, 14-07-2004		
Jenis Kelamin	: LAKI-LAKI	Gol.Darah : B	
Alamat	: GLAGAH UH 4/67		
RT/RW	: 012/003		
Kel/Desa	: WARUNGBOTO		
Kecamatan	: UMBULHARJO		
Agama	: ISLAM		
Status Perkawinan	: BELUM KAWIN	Foto.jpg	
Pekerjaan	: PELAJAR/MAHASISWA		
Kewarganegaraan	: WNI	24-06-2021	
Berlaku Hingga	: SEUMUR HIDUP	TTD	
=====			
PROVINSI DAERAH ISTIMEWA YOGYAKARTA			
KOTA YOGYAKARTA			
NIK	: 3203012503770011		
Nama	: GUOHUI CHEN		
Tempat/Tgl Lahir	: FUJIAN, 25-03-1977		
Jenis Kelamin	: PEREMPUAN	Gol.Darah : A	
Alamat	: JLN. GELAMET PERUMAHAN BANGSA RT. NO. 40		

Akan mencoba delete di awal (head) menggunakan doubly linked list

```
Menu:
1. Delete data di depan
2. Delete data di belakang
3. Kembali
Pilihan: 1
Data berhasil dihapus
Press any key to continue . . .
```

=====			
PROVINSI KALIMANTAN TENGAH			
KABUPATEN KOTAWARINGIN BARAT			
NIK	:	3471135407040001	
Nama	:	AISYAH SYAFI'I NURJANNAH	
Tempat/Tgl Lahir	:	YOGYAKARTA, 14-07-2004	
Jenis Kelamin	:	LAKI-LAKI	Gol.Darah : B
Alamat	:	GLAGAH UH 4/67	
RT/RW	:	012/003	
Kel/Desa	:	WARUNGBOTO	
Kecamatan	:	UMBULHARJO	
Agama	:	ISLAM	
Status Perkawinan	:	BELUM KAWIN	Foto.jpg
Pekerjaan	:	PELAJAR/MAHASISWA	
Kewarganegaraan	:	WNI	24-06-2021
Berlaku Hingga	:	SEUMUR HIDUP	TTD
=====			
PROVINSI DAERAH ISTIMEWA YOGYAKARTA			
KOTA YOGYAKARTA			
NIK	:	3203012503770011	
Nama	:	GUOHUI CHEN	
Tempat/Tgl Lahir	:	FUJIAN, 25-03-1977	
Jenis Kelamin	:	PEREMPUAN	Gol.Darah : A
Alamat	:	JL. SELAMET PERUMAHAN RANCABALI NO.40	
RT/RW	:	002/004	
Kel/Desa	:	MUKA	
Kecamatan	:	CIANJUR	
Agama	:	KRISTEN	
Status Perkawinan	:	KAWIN	Foto.jpg
Pekerjaan	:	OTHERS	
Kewarganegaraan	:	WNI	11-10-2023
Berlaku Hingga	:	SEUMUR HIDUP	TTD
=====			
PROVINSI DKI JAKARTA			
JAKARTA BARAT			
NIK	:	3171234567890123	
Nama	:	MIRA SETIAWAN	

Data paling awal KTP bernama Mohammad Farid Hendianto sudah berhasil di hapus



Kewarganegaraan	: WNI	17-10-2023
Berlaku Hingga	: SEUMUR HIDUP	TTD
=====		
PROVINSI JAWA TENGAH		
KABUPATEN KUDUS		
NIK	: 3319042612640001	
Nama	: BUDI SANTOSO	
Tempat/Tgl Lahir	: KUDUS, 29-12-1984	
Jenis Kelamin	: PEREMPUAN	Gol.Darah : B
Alamat	: LARIREJO	
RT/RW	: 002/001	
Kel/Desa	: LARIREJO	
Kecamatan	: UNDAAN	
Agama	: ISLAM	
Status Perkawinan	: BELUM KAWIN	Foto.jpg
Pekerjaan	: PETANI/PEKEBUN	
Kewarganegaraan	: WNI	17-10-2023
Berlaku Hingga	: SEUMUR HIDUP	TTD
=====		
PROVINSI JAWA TENGAH		
KOTA SURAKARTA		
NIK	: 3372011210770001	
Nama	: SELAMAT IDUL ADHA	
Tempat/Tgl Lahir	: SURAKARTA, 12-10-1977	
Jenis Kelamin	: PEREMPUAN	Gol.Darah : AB
Alamat	: BARON CILIK	
RT/RW	: 004/005	
Kel/Desa	: BUMI	
Kecamatan	: LAWEYAN	
Agama	: ISLAM	
Status Perkawinan	: KAWIN	Foto.jpg
Pekerjaan	: PERDAGANGAN	
Kewarganegaraan	: WNI	17-10-2023
Berlaku Hingga	: SEUMUR HIDUP	TTD
=====		
PROVINSI BANTEN		
KABUPATEN PANDEGLANG		
NIK	: 3601120705880002	
Nama	: RUSLI	
Tempat/Tgl Lahir	: PANDEGLANG, 17-07-1988	
Jenis Kelamin	: PEREMPUAN	Gol.Darah : A
Alamat	: KP BADONGAN	
RT/RW	: 001/013	
Kel/Desa	: TELUK	
Kecamatan	: LABUAN	
Agama	: ISLAM	
Status Perkawinan	: BELUM KAWIN	Foto.jpg
Pekerjaan	: WIRASWASTA	
Kewarganegaraan	: WNI	31-10-2023

Akan mencoba menghapus data Selamat Idul Adha dan Rusli, nanti yang bagian tail hanya tersisa dari Budi Santoso.

```
Menu:
1. Delete data di depan
2. Delete data di belakang
3. Kembali
Pilihan: 2
Data berhasil dihapus
Press any key to continue
```

Sudah mendelete sebanyak 2 kali.

KABUPATEN PEKALONGAN			
NIK	: 3329091003780012		
Nama	: AMAAT FAOZI		
Tempat/Tgl Lahir	: PEKALONGAN, 10-03-1978		
Jenis Kelamin	: PEREMPUAN	Gol.Darah : A	
Alamat	: DUSUN KAUMAN		
RT/Rw	: 002/005		
Kel/Desa	: KESESI		
Kecamatan	: KESESI		
Agama	: ISLAM		
Status Perkawinan	: KAWIN	Foto.jpg	
Pekerjaan	: Pedagang		
Kewarganegaraan	: WNI	17-10-2023	
Berlaku Hingga	: SEUMUR HIDUP	TTD	
=====			
PROVINSI JAWA TENGAH			
KABUPATEN KUDUS			
NIK	: 3319042612640001		
Nama	: BUDI SANTOSO		
Tempat/Tgl Lahir	: KUDUS, 29-12-1984		
Jenis Kelamin	: PEREMPUAN	Gol.Darah : B	
Alamat	: LARIREJO		
RT/Rw	: 002/001		
Kel/Desa	: LARIREJO		
Kecamatan	: UNDAAN		
Agama	: ISLAM		
Status Perkawinan	: BELUM KAWIN	Foto.jpg	
Pekerjaan	: PETANI/PEKEBUN		
Kewarganegaraan	: WNI	17-10-2023	
Berlaku Hingga	: SEUMUR HIDUP	TTD	
=====			

Yang terakhir akan yang sebagai tail adalah Budi Santoso.

Operasi menambah data

Sudah jelas pada bagian kelas KTP dibagian prosedur input(). Penambahan yang digunakan adalah dengan konsep Doubly Linked List Insert at Tail

Operasi menghapus data

Sudah jelas pada bagian kelas KTP dibagian prosedur delAtHead() untuk mendelete data dengan konsep Doubly Linked List dari bagian head, dan delAtTail() untuk mendelete data dengan konsep Doubly Linked List dari bagian tail.

Class

Pada progra sudah menggunakan class KTP dengan objek instance ktp.

Penjelasan lebih lanjut

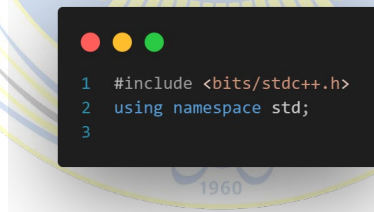
Sebelumnya, kodingan menggunakan linked list tunggal untuk menyimpan data KTP. Sedangkan pada kodingan sesudahnya, digunakan doubly linked list untuk menyimpan data KTP.

Pada kodingan sebelumnya, struktur data linked list tunggal didefinisikan menggunakan struct Node yang memiliki dua atribut yaitu data yang merupakan objek dari struct Data dan next yang merupakan pointer ke node selanjutnya dalam linked list. Kelas KTP memiliki atribut head yang merupakan pointer ke node pertama dalam linked list dan atribut count yang menyimpan jumlah node dalam linked list.

Pada kodingan sesudahnya, struktur data doubly linked list didefinisikan menggunakan struct Node yang memiliki tiga atribut yaitu data yang merupakan objek dari struct Data, next yang merupakan pointer ke node selanjutnya dalam linked list, dan prev yang merupakan pointer ke node sebelumnya dalam linked list. Kelas KTP memiliki atribut head yang merupakan pointer ke node pertama dalam linked list, atribut tail yang merupakan pointer ke node terakhir dalam linked list, dan atribut count yang menyimpan jumlah node dalam linked list.

Perbedaan utama antara linked list tunggal dan doubly linked list terletak pada kemampuan doubly linked list untuk menavigasi ke node sebelumnya dalam linked list. Dalam linked list tunggal, kita hanya dapat mengakses node selanjutnya, sedangkan dalam doubly linked list, kita dapat mengakses node selanjutnya dan sebelumnya.

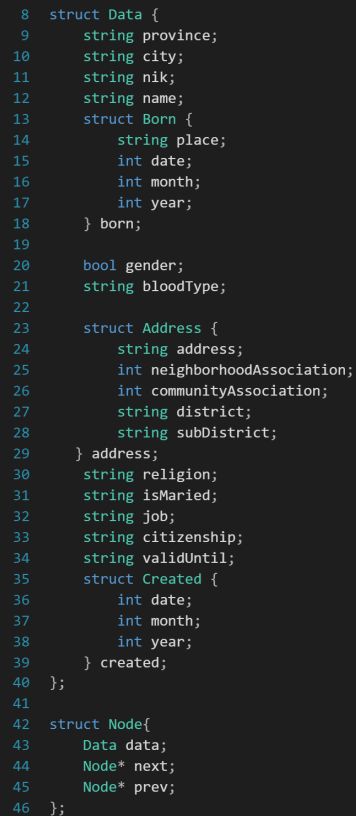
Berikut adalah jelasan kodingan per bagian.



Header yang lebih ringkas

Kode tersebut merupakan header file yang memuat semua header file standar dari C++ dan digunakan untuk mempersingkat penulisan kode. Header file ini biasanya digunakan pada kompetisi pemrograman atau saat ingin menulis kode dengan cepat tanpa harus menuliskan satu per satu header file yang dibutuhkan.

Kode using namespace std; digunakan untuk menghindari penulisan std:: pada setiap penggunaan fungsi atau objek dari namespace std. Namun, penggunaan using namespace std; tidak disarankan pada kode besar karena dapat menimbulkan konflik nama (name collision) pada namespace yang berbeda.



```

8  struct Data {
9      string province;
10     string city;
11     string nik;
12     string name;
13     struct Born {
14         string place;
15         int date;
16         int month;
17         int year;
18     } born;
19
20     bool gender;
21     string bloodType;
22
23     struct Address {
24         string address;
25         int neighborhoodAssociation;
26         int communityAssociation;
27         string district;
28         string subDistrict;
29     } address;
30     string religion;
31     string isMarried;
32     string job;
33     string citizenship;
34     string validUntil;
35     struct Created {
36         int date;
37         int month;
38         int year;
39     } created;
40 };
41
42 struct Node{
43     Data data;
44     Node* next;
45     Node* prev;
46 };

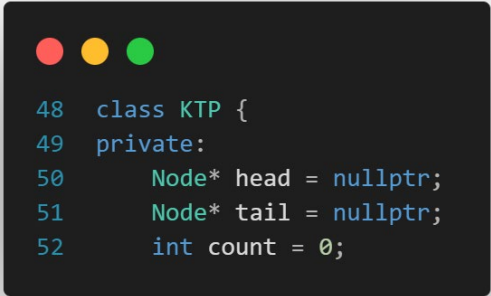
```

Inisialisasi struct data dan KTP

Kode tersebut mendefinisikan dua buah struktur data yaitu Data dan Node. Struktur Data memiliki beberapa variabel anggota (member variables) seperti province, city, nik, name, born, gender, bloodType, address, religion, isMarried, job, citizenship, validUntil, dan created.

Variabel born dan created merupakan struktur data lagi yang memiliki beberapa variabel anggota seperti place, date, month, dan year. Variabel address juga merupakan struktur data lagi yang memiliki beberapa variabel anggota seperti address, neighborhoodAssociation, communityAssociation, district, dan subDistrict.

Struktur Node memiliki tiga variabel anggota yaitu data, next, dan prev. Variabel data merupakan objek dari struktur Data, sedangkan variabel next dan prev merupakan pointer ke objek Node selanjutnya dan sebelumnya dalam suatu linked list.



```
48 class KTP {  
49 private:  
50     Node* head = nullptr;  
51     Node* tail = nullptr;  
52     int count = 0;
```

Inisialisasi head dan tail, inisialisasi class KTP

Kode tersebut mendefinisikan sebuah kelas KTP yang memiliki tiga variabel anggota head, tail, dan count. Variabel head dan tail merupakan pointer ke objek Node pertama dan terakhir dalam suatu linked list. Variabel count merupakan jumlah elemen dalam linked list.

Kelas KTP memiliki akses modifier private yang artinya variabel anggota hanya dapat diakses dari dalam kelas KTP itu sendiri. Hal ini bertujuan untuk membatasi akses langsung ke variabel anggota dari luar kelas dan mencegah perubahan yang tidak diinginkan.


```

53
54 public:
55     bool validateProvince(string province) {
56         return province.length() <= 50;
57     }
58
59     bool validateCity(string city) {
60         return city.length() <= 50;
61     }
62
63
64     bool validateNik(string nik) {
65         return nik.length() == 16;
66     }
67
68     bool validateName(string name) {
69         return name.length() <= 50;
70     }
71
72     bool validateBornPlace(string bornPlace) {
73         return bornPlace.length() <= 20;
74     }
75
76     bool validateDate(int date) {
77         return date >= 1 && date <= 31;
78     }
79
80     bool validateMonth(int month) {
81         return month >= 1 && month <= 12;
82     }
83
84     bool validateYear(int year) {
85         return year >= 1900 && year <= 2023;
86     }
87
88     bool validateGender(int gender) {
89         return gender == 0 || gender == 1;
90     }
91
92     bool validateBloodType(string bloodType) {
93         return bloodType == "A" || bloodType == "B" || bloodType == "AB" || bloodType == "O";
94     }
95
96     bool validateAddress(string address) {
97         return address.length() <= 100;
98     }
99
100     bool validateNeighborhoodAssociation(int neighborhoodAssociation) {
101         return neighborhoodAssociation >= 0 && neighborhoodAssociation <= 999;
102     }
103
104     bool validateCommunityAssociation(int communityAssociation) {
105         return communityAssociation >= 0 && communityAssociation <= 999;
106     }
107
108     bool validateSubDistrict(string subDistrict) {
109         return subDistrict.length() <= 20;
110     }
111
112     bool validateDistrict(string district) {
113         return district.length() <= 20;
114     }
115
116     bool validateReligion(string religion) {
117         return religion == "Islam" || religion == "Kristen" || religion == "Katolik" || religion == "Hindu" || religion == "Buddha" || religion == "Konghucu";
118     }
119
120     bool validateIsMarried(string isMarried) {
121         return isMarried == "Kawin" || isMarried == "Belum Kawin";
122     }
123
124     bool validateJob(string job) {
125         return job.length() <= 30;
126     }
127

```

Kode tersebut merupakan implementasi dari beberapa fungsi validasi untuk memeriksa apakah data yang dimasukkan sesuai dengan format yang diinginkan.

Fungsi validateProvince, validateCity, validateName, validateBornPlace, validateBloodType, validateAddress, validateSubDistrict, validateDistrict, validateReligion, dan validateJob memeriksa apakah panjang string yang dimasukkan tidak melebihi batas maksimum yang ditentukan.

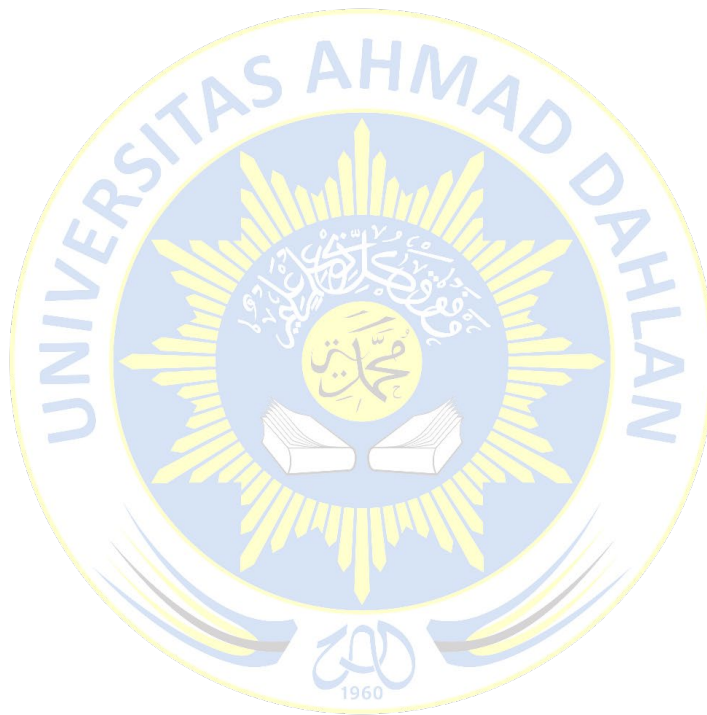
Fungsi validateNik memeriksa apakah panjang string nik sama dengan 16.

Fungsi validateDate, validateMonth, dan validateYear memeriksa apakah tanggal, bulan, dan tahun yang dimasukkan berada dalam rentang yang valid.

Fungsi `validateGender` memeriksa apakah nilai gender yang dimasukkan hanya bernilai 0 atau 1.

Fungsi `validateNeighborhoodAssociation` dan `validateCommunityAssociation` memeriksa apakah nilai `neighborhoodAssociation` dan `communityAssociation` yang dimasukkan berada dalam rentang yang valid.

Fungsi `validateIsMaried` memeriksa apakah nilai `isMaried` yang dimasukkan hanya bernilai "Kawin" atau "Belum Kawin".



Pertemuan ke-5: POST TEST – Variasi Link List

Prosedur input() pada class KTP

Kode tersebut merupakan implementasi dari fungsi input yang bertujuan untuk memasukkan data ke dalam linked list dan menyimpannya ke dalam file data.txt.

Pada bagian `Node* head = nullptr;` dan `Node* tail = nullptr;`, variabel `head` dan `tail` dideklarasikan sebagai pointer ke objek `Node` yang awalnya diinisialisasi dengan nilai `nullptr`.

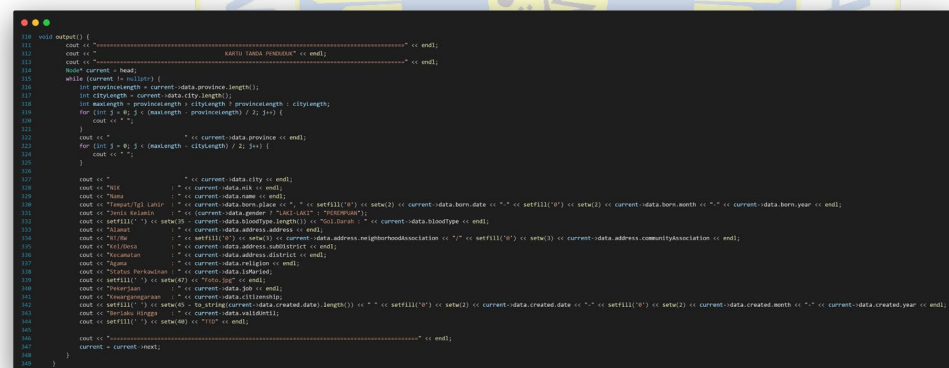
Pada bagian `newNode->next = nullptr;`, variabel `next` dari objek `newNode` diinisialisasi dengan nilai `nullptr`.

Pada bagian `if (head == nullptr) { ... }`, jika linked list masih kosong, maka `head` dan `tail` diisi dengan objek `newNode` dan variabel `prev` dari objek `newNode` diinisialisasi dengan nilai `nullptr`.

Pada bagian `else { ... }`, jika linked list tidak kosong, maka variabel `next` dari objek `tail` diisi dengan objek `newNode`, variabel `prev` dari objek `newNode` diisi dengan objek `tail`, dan variabel `tail` diisi dengan objek `newNode`.

Pada bagian `count++;`, variabel `count` diincrement setiap kali objek baru berhasil dimasukkan ke dalam linked list.

Setelah data berhasil dimasukkan ke dalam linked list, data juga akan disimpan ke dalam file `data.txt` dengan format yang telah ditentukan.



```

300 void output() {
301     cout << "=====KTP===== " << endl;
302     cout << "data data KTP " << endl;
303     cout << "===== " << endl;
304     Node* current = head;
305     while (current != nullptr) {
306         int provinceLength = current->data.province.length();
307         int cityLength = current->data.city.length();
308         int provinceLength + cityLength = provinceLength + cityLength;
309         for (int i = 0; i < (provinceLength + cityLength) / 2; i++) {
310             cout << " ";
311         }
312         cout << " " << current->data.province << endl;
313         for (int j = 0; j < (provinceLength + cityLength) / 2; j++) {
314             cout << " ";
315         }
316         cout << " " << current->data.city << endl;
317         cout << "NIK " << current->data.nik << endl;
318         cout << "Nama " << current->data.name << endl;
319         cout << "Tempat/tgl lahir " << current->data.born.place << " " << setfill('0') << setw(2) << current->data.born.month << " " << current->data.born.year << endl;
320         cout << "Jenis kelamin " << current->data.gender << " " << setfill('0') << setw(2) << current->data.bloodType << endl;
321         cout << setfill(' ') << setw(8) << current->data.bloodType.length() << " " << current->data.bloodType << endl;
322         cout << "Alamat " << current->data.address.address << endl;
323         cout << "RT/RW " << setfill(' ') << setw(2) << current->data.address.addressCommunityAssociation << setfill('0') << setw(2) << current->data.address.addressCommunityAssociation << endl;
324         cout << "Kelurahan " << current->data.address.district << endl;
325         cout << "Kecamatan " << current->data.address.district << endl;
326         cout << "Agama " << current->data.religion << endl;
327         cout << "Status Perkawinan " << current->data.married << endl;
328         cout << setfill(' ') << setw(4) << "Photo: " << endl;
329         cout << "Photo: " << current->data.photo << endl;
330         cout << "Masa berlaku " << current->data.validity << endl;
331         cout << setfill(' ') << setw(4) << "Created: " << endl;
332         cout << "Created: " << current->data.created.data.length() << " " << setfill('0') << setw(2) << current->data.created.month << " " << setfill('0') << setw(2) << current->data.created.year << endl;
333         cout << "Berlaku hingga " << current->data.validity << endl;
334         cout << setfill(' ') << setw(4) << "Valid: " << endl;
335         cout << "Valid: " << current->data.validity << endl;
336         cout << "===== " << endl;
337         current = current->next;
338     }
339 }

```

Prosedur output()

Ini adalah sebuah fungsi yang digunakan untuk mencetak data dari linked list yang terhubung dua arah (doubly linked list). Fungsi ini mencetak data dari setiap node pada linked list, dimulai dari `head` hingga `tail`. Setiap node memiliki beberapa data seperti provinsi, kota, NIK, nama, tempat/tanggal lahir, jenis kelamin, golongan darah, alamat, RT/RW, kelurahan/desa, kecamatan, agama, status perkawinan, foto, pekerjaan, kewarganegaraan, tanggal pembuatan, dan masa berlaku. Data-data ini dicetak dengan format yang sudah ditentukan.

Pertama-tama, fungsi ini mencetak header yang terdiri dari tiga baris. Kemudian, fungsi ini menginisialisasi variabel current dengan head. Selanjutnya, fungsi ini melakukan looping untuk setiap node pada linked list. Pada setiap iterasi, fungsi ini menghitung panjang provinsi dan kota, dan menentukan panjang maksimum dari keduanya. Kemudian, fungsi ini mencetak provinsi dan kota dengan format tertentu, diikuti oleh data-data lainnya seperti NIK, nama, tempat/tanggal lahir, jenis kelamin, golongan darah, alamat, RT/RW, kelurahan/desa, kecamatan, agama, status perkawinan, foto, pekerjaan, kewarganegaraan, tanggal pembuatan, dan masa berlaku. Setelah mencetak data dari suatu node, fungsi ini menggeser current ke node berikutnya.

```

351 void initialize() {
352     ifstream file;
353     file.open("data.txt");
354
355     string line;
356     count = 0;
357     while (getline(file, line)) {
358         if (line.find("NIK") != string::npos) {
359             count++;
360         }
361     }
362
363     cout << "Banyak KTP: " << count << endl;
364     file.clear(); file.seekg(0, ios::beg);
365
366     this->head = nullptr;
367     Node* tail = nullptr;
368
369     for (int i = 0; i < count; i++) {
370         Data newData;
371
372         while (getline(file, line)) {
373             if (line.find("Provinsi") != string::npos) newData.province = line.substr(line.find(":") + 2);
374             else if (line.find("Kota/Kabupaten") != string::npos) newData.city = line.substr(line.find(":") + 2);
375             else if (line.find("NIK") != string::npos) newData.nik = line.substr(line.find(":") + 2);
376             else if (line.find("Nama") != string::npos) newData.name = line.substr(line.find(":") + 2);
377             else if (line.find("Tempat/Tgl Lahir") != string::npos) {
378                 newData.born.place = line.substr(line.find(":") + 2, line.find(",") - line.find(":") - 2);
379                 newData.born.date = stoi(line.substr(line.find(",") + 2, 2));
380                 newData.born.month = stoi(line.substr(line.find("-") + 1, 2));
381                 newData.born.year = stoi(line.substr(line.find_last_of("-") + 1));
382             }
383             else if (line.find("Jenis Kelamin") != string::npos) {
384                 newData.gender = line.substr(line.find(":") + 2) == "LAKI-LAKI" ? 0 : 1;
385             }
386             else if (line.find("Gol.Darah") != string::npos) newData.bloodType = line.substr(line.find(":") + 2);
387             else if (line.find("Alamat") != string::npos) newData.address.address = line.substr(line.find(":") + 2);
388             else if (line.find("RT/RW") != string::npos) {
389                 newData.address.neighborhoodAssociation = stoi(line.substr(line.find(":") + 2, 3));
390                 newData.address.communityAssociation = stoi(line.substr(line.find_last_of("/") + 1));
391             }
392             else if (line.find("Kel.Desa") != string::npos) newData.address.subDistrict = line.substr(line.find(":") + 2);
393             else if (line.find("Kecamatan") != string::npos) newData.address.district = line.substr(line.find(":") + 2);
394             else if (line.find("Agama") != string::npos) newData.religion = line.substr(line.find(":") + 2);
395             else if (line.find("Status Perkawinan") != string::npos) newData.isMarried = line.substr(line.find(":") + 2);
396             else if (line.find("Pekerjaan") != string::npos) newData.job = line.substr(line.find(":") + 2);
397             else if (line.find("Kewarganegaraan") != string::npos) newData.citizenship = line.substr(line.find(":") + 2);
398             else if (line.find("Berlaku Hingga") != string::npos) newData.validUntil = line.substr(line.find(":") + 2);
399             else if (line.find("Dibuat") != string::npos) {
400                 newData.created.date = stoi(line.substr(line.find(":") + 2, 2));
401                 newData.created.month = stoi(line.substr(line.find("-") + 1, 2));
402                 newData.created.year = stoi(line.substr(line.find_last_of("-") + 1));
403             }
404             else if (line.find("=====") != string::npos) {
405                 Node* newNode = new Node( newData, nullptr, tail );
406                 if (head == nullptr) {
407                     head = newNode;
408                     tail = newNode;
409                 }
410                 else {
411                     tail->next = newNode;
412                     tail = newNode;
413                 }
414                 break;
415             }
416         }
417     }
418
419     file.close();
420 }

```

Prosedur initialize() pada class KTP

Ini adalah sebuah fungsi yang digunakan untuk membaca data dari file "data.txt" dan memasukkannya ke dalam linked list yang terhubung dua arah (doubly linked list). Fungsi ini menghitung jumlah data yang ada pada file dengan mencari setiap baris yang mengandung kata "NIK". Setelah itu, fungsi membuka kembali file dan membaca setiap barisnya. Setiap baris

yang mengandung informasi KTP akan diproses dan dimasukkan ke dalam sebuah objek Data. Setelah semua informasi KTP pada satu objek Data terkumpul, objek Data tersebut akan dimasukkan ke dalam sebuah node baru pada linked list.

Pertama-tama, fungsi ini membuka file "data.txt" dan menghitung jumlah data yang ada pada file dengan melakukan looping pada setiap baris dan mencari setiap baris yang mengandung kata "NIK". Setelah itu, fungsi ini mencetak jumlah data yang ditemukan. Kemudian, fungsi ini menginisialisasi head dan tail dari linked list dengan nullptr.

Selanjutnya, fungsi ini melakukan looping sebanyak jumlah data yang ditemukan pada file. Pada setiap iterasi, fungsi ini membuat sebuah objek Data baru dan melakukan looping pada setiap baris pada file. Setiap baris yang mengandung informasi KTP akan diproses dan dimasukkan ke dalam objek Data yang baru dibuat. Setelah semua informasi KTP pada satu objek Data terkumpul, objek Data tersebut akan dimasukkan ke dalam sebuah node baru pada linked list.

Setelah semua data pada file telah diproses, fungsi ini menutup file "data.txt".



```

421
422 void delAtHead(){
423     if(head == nullptr){
424         cout << "Data kosong" << endl;
425         return;
426     }
427     Node* temp = head;
428     head = head->next;
429     if(head != nullptr){
430         head->prev = nullptr;
431     }
432     delete temp;
433     cout << "Data berhasil dihapus" << endl;
434     count--;
435
436     ifstream file;
437     file.open("data.txt");
438     ofstream tempFile;
439     tempFile.open("temp.txt");
440     string line;
441     int lineCount = 0;
442     while (getline(file, line)) {
443         if (line.find("NIK") != string::npos) {
444             lineCount++;
445         }
446         if (lineCount != 1) {
447             tempFile << line << endl;
448         }
449     }
450     file.close();
451     tempFile.close();
452     remove("data.txt");
453     rename("temp.txt", "data.txt");
454 }
455
456 void delAtTail(){
457     if(head == nullptr){
458         cout << "Linked list kosong" << endl;
459         return;
460     }
461     Node* current = head;
462     while(current->next != nullptr){
463         current = current->next;
464     }
465     if(current->prev == nullptr){
466         head = nullptr;
467     } else {
468         current->prev->next = nullptr;
469     }
470     delete current;
471     cout << "Data berhasil dihapus" << endl;
472     count--;
473
474     ifstream file;
475     file.open("data.txt");
476     ofstream tempFile;
477     tempFile.open("temp.txt");
478     string line;
479     int lineCount = 0;
480     while (getline(file, line)) {
481         if (line.find("NIK") != string::npos) {
482             lineCount++;
483         }
484         if (lineCount != count + 1) {
485             tempFile << line << endl;
486         }
487     }
488     file.close();
489     tempFile.close();
490     remove("data.txt");
491     rename("temp.txt", "data.txt");
492 }
493 };

```

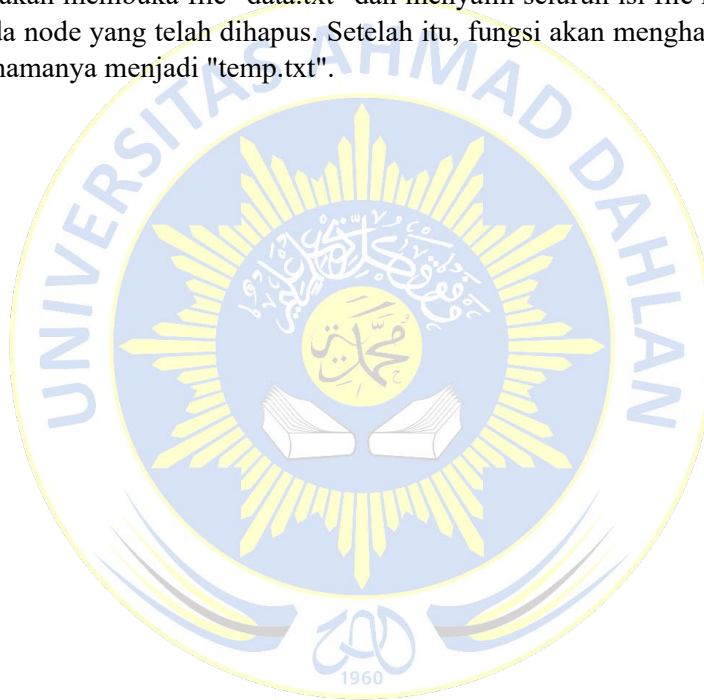
delAtHead() dan delAtTail() doubly linked list di class KTP

Ini adalah dua buah fungsi yang digunakan untuk menghapus node pada linked list yang terhubung dua arah (doubly linked list). Fungsi delAtHead() digunakan untuk menghapus node pada bagian depan linked list, sedangkan fungsi delAtTail() digunakan untuk menghapus node pada bagian belakang linked list.

Fungsi delAtHead() akan menghapus node pertama pada linked list. Pertama-tama, fungsi ini akan memeriksa apakah linked list kosong atau tidak. Jika linked list kosong, maka fungsi akan mencetak pesan "Data kosong" dan langsung keluar dari fungsi. Jika linked list

tidak kosong, maka fungsi akan menghapus node pertama pada linked list dengan mengubah pointer head dan prev dari node-node terkait. Setelah node berhasil dihapus, fungsi akan membuka file "data.txt" dan menyalin seluruh isi file ke file "temp.txt", kecuali data pada node yang telah dihapus. Setelah itu, fungsi akan menghapus file "data.txt" dan mengganti namanya menjadi "temp.txt".

Fungsi delAtTail() akan menghapus node terakhir pada linked list. Pertama-tama, fungsi ini akan memeriksa apakah linked list kosong atau tidak. Jika linked list kosong, maka fungsi akan mencetak pesan "Linked list kosong" dan langsung keluar dari fungsi. Jika linked list tidak kosong, maka fungsi akan mencari node terakhir pada linked list dengan melakukan looping pada setiap node. Setelah node terakhir ditemukan, fungsi akan menghapus node tersebut dengan mengubah pointer next dan prev dari node-node terkait. Setelah node berhasil dihapus, fungsi akan membuka file "data.txt" dan menyalin seluruh isi file ke file "temp.txt", kecuali data pada node yang telah dihapus. Setelah itu, fungsi akan menghapus file "data.txt" dan mengganti namanya menjadi "temp.txt".



```

495 int main() {
496     KTP ktp;
497     int choice;
498     while(true){
499         ktp.initialize();
500         cout << "Menu: " << endl;
501         cout << "1. Input Data" << endl;
502         cout << "2. Tampilkan Data" << endl;
503         cout << "3. Delete Data" << endl;
504         cout << "4. Keluar" << endl;
505         cout << "Pilihan: ";
506         cin >> choice;
507         system("cls");
508         cin.ignore();
509         switch (choice) {
510             case 1:
511                 ktp.input();
512                 cout << "Data berhasil diinput" << endl;
513                 break;
514             case 2:
515                 ktp.output();
516                 break;
517             case 3:
518                 while(true){
519                     system("cls");
520                     cout << "Menu: " << endl;
521                     cout << "1. Delete data di depan" << endl;
522                     cout << "2. Delete data di belakang" << endl;
523                     cout << "3. Kembali" << endl;
524                     cout << "Pilihan: ";
525                     cin >> choice;
526                     switch(choice){
527                         case 1:
528                             ktp.delAtHead();
529                             break;
530                         case 2:
531                             ktp.delAtTail();
532                             break;
533                         case 3:
534                             break;
535                         default:
536                             cout << "Pilihan tidak valid" << endl;
537                             system("pause");
538                             break;
539                     }
540                 }
541             case 4:
542                 break;
543             default:
544                 return EXIT_SUCCESS;
545             case 4:
546                 cout << "Pilihan tidak valid" << endl;
547                 break;
548         }
549         system("pause");
550         system("cls");
551     }
552 }

```

Fungsi utama

Ini adalah sebuah program utama yang digunakan untuk menjalankan aplikasi KTP. Program ini menggunakan objek KTP yang telah didefinisikan sebelumnya. Program ini memiliki empat pilihan menu: input data, tampilkan data, hapus data, dan keluar.

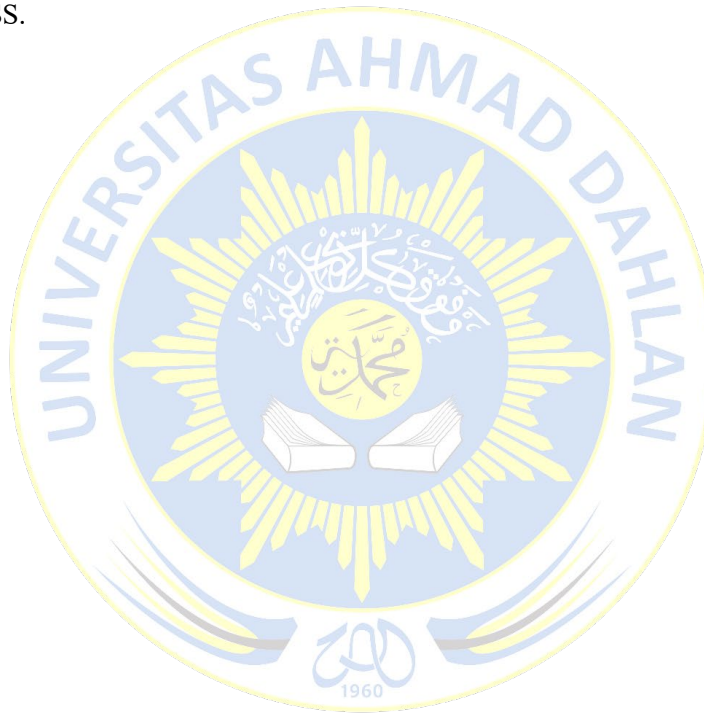
Pada awal program, objek KTP diinisialisasi dengan memanggil fungsi initialize(). Setelah itu, program akan menampilkan menu dan meminta pengguna untuk memilih salah satu pilihan. Setelah pengguna memilih pilihan, program akan memanggil fungsi yang sesuai dengan pilihan tersebut.

Jika pengguna memilih pilihan 1, program akan memanggil fungsi `input()` dari objek KTP untuk meminta pengguna memasukkan data KTP baru. Setelah data dimasukkan, program akan mencetak pesan "Data berhasil diinput".

Jika pengguna memilih pilihan 2, program akan memanggil fungsi `output()` dari objek KTP untuk menampilkan seluruh data KTP yang telah dimasukkan sebelumnya.

Jika pengguna memilih pilihan 3, program akan menampilkan submenu yang meminta pengguna untuk memilih apakah data yang ingin dihapus berada di depan atau di belakang linked list. Setelah pengguna memilih, program akan memanggil fungsi `delAtHead()` atau `delAtTail()` dari objek KTP untuk menghapus data yang dipilih.

Jika pengguna memilih pilihan 4, program akan keluar dengan mengembalikan nilai `EXIT_SUCCESS`.



```

495 int main() {
496     KTP ktp;
497     int choice;
498     while(true){
499         ktp.initialize();
500         cout << "Menu: " << endl;
501         cout << "1. Input Data" << endl;
502         cout << "2. Tampilkan Data" << endl;
503         cout << "3. Delete Data" << endl;
504         cout << "4. Keluar" << endl;
505         cout << "Pilihan: ";
506         cin >> choice;
507         system("cls");
508         cin.ignore();
509         switch (choice) {
510             case 1:
511                 ktp.input();
512                 cout << "Data berhasil diinput" << endl;
513                 break;
514             case 2:
515                 ktp.output();
516                 break;
517             case 3:
518                 while(true){
519                     system("cls");
520                     cout << "Menu: " << endl;
521                     cout << "1. Delete data di depan" << endl;
522                     cout << "2. Delete data di belakang" << endl;
523                     cout << "3. Kembali" << endl;
524                     cout << "Pilihan: ";
525                     cin >> choice;
526                     switch(choice){
527                         case 1:
528                             ktp.delAtHead();
529                             break;
530                         case 2:
531                             ktp.delAtTail();
532                             break;
533                         case 3:
534                             break;
535                         default:
536                             cout << "Pilihan tidak valid" << endl;
537                             system("pause");
538                             break;
539                     }
540                     break;
541                 }
542             case 4:
543                 return EXIT_SUCCESS;
544             default:
545                 cout << "Pilihan tidak valid" << endl;
546                 break;
547         }
548         system("pause");
549         system("cls");
550     }
551 }
552 }
553 }

```

Untuk mengakses kodingan, dapat melihat link github berikut:

[IRedDragonICY/Data-Structure \(github.com\)](https://github.com/IRedDragonICY/Data-Structure)

