

LAPORAN PRAKTIKUM

“Pertemuan ke-4: POST TEST – 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**

```

1 #include <iostream>
2 #include <string>
3 #include <iomanip>
4 #include <iomanip>
5 #include <iomanip>
6 using namespace std;
7
8 void errorInput() {
9     cout << "Input data valid" << endl;
10 }
11
12 struct Data {
13     string province;
14     string city;
15     string nia;
16     string name;
17     struct Born {
18         string place;
19         int date;
20         int month;
21         int year;
22     } born;
23
24     bool gender;
25     string bloodtype;
26
27     struct Address {
28         string address;
29         int neighborhoodAssociation;
30         int communityAssociation;
31         string district;
32         string subDistrict;
33     } address;
34     string religion;
35     string married;
36     string job;
37     string citizenship;
38     string validated;
39     struct Created {
40         int date;
41         int month;
42         int year;
43     } created;
44 };
45
46 struct Node {
47     Data data;
48     Node* next;
49 };
50
51 class MTP {
52 private:
53     Node* head = nullptr;
54     int count = 0;
55 public:
56
57     bool validateProvince(string province) {
58         return province.length() <= 50;
59     }
60
61     bool validateCity(string city) {
62         return city.length() <= 50;
63     }
64
65     bool validateNia(string nia) {
66         return nia.length() <= 20;
67     }
68     bool validateName(string name) {
69         return name.length() <= 50;
70     }
71     bool validateBornPlace(string bornPlace) {
72         return bornPlace.length() <= 20;
73     }
74
75     bool validateDate(int date) {
76         return date >= 1 && date <= 31;
77     }
78
79     bool validateMonth(int month) {
80         return month >= 1 && month <= 12;
81     }
82
83     bool validateYear(int year) {
84         return year >= 1900 && year <= 2023;
85     }
86
87     bool validateGender(int gender) {
88         return gender == 0 || gender == 1;
89     }
90
91     bool validateBloodType(string bloodtype) {
92         return bloodtype == "A" || bloodtype == "B" || bloodtype == "AB" || bloodtype == "O";
93     }
94
95     bool validateAddress(string address) {
96         return address.length() <= 100;
97     }
98
99     bool validateNeighborhoodAssociation(int neighborhoodAssociation) {
100         return neighborhoodAssociation >= 0 && neighborhoodAssociation <= 999;
101     }
102
103     bool validateCommunityAssociation(int communityAssociation) {
104         return communityAssociation >= 0 && communityAssociation <= 999;
105     }
106
107     bool validateSubDistrict(string subDistrict) {
108         return subDistrict.length() <= 20;
109     }
110
111     bool validateDistrict(string district) {
112         return district.length() <= 20;
113     }
114
115     bool validateReligion(string religion) {
116         return religion == "Krisna" || religion == "Katolia" || religion == "Muda" || religion == "Buddha" || religion == "Manghuca";
117     }
118
119     bool validateMarried(string married) {
120         return married == "Kawin" || married == "Belum Kawin";
121     }
122
123     bool validateJob(string job) {
124         return job.length() <= 20;
125     }
126
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 << "City: ";
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 << "Nia: ";
145             getline(cin, newnode->data.nia);
146             if (validateNia(newnode->data.nia)) {
147                 errorInput();
148             }
149         } while (!validateNia(newnode->data.nia));
150         do {
151             cout << "Name: ";
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 << "Born Place: ";
159             getline(cin, newnode->data.bornPlace);
160             if (validateBornPlace(newnode->data.bornPlace)) {
161                 errorInput();
162             }
163         } while (!validateBornPlace(newnode->data.bornPlace));
164         do {
165             cout << "Date: ";
166             getline(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 << "Month: ";
173             getline(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 << "Year: ";
180             getline(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 << "Gender: ";
187             getline(cin, newnode->data.gender);
188             if (validateGender(newnode->data.gender)) {
189                 errorInput();
190             }
191         } while (!validateGender(newnode->data.gender));
192         do {
193             cout << "Blood Type: ";
194             getline(cin, newnode->data.bloodtype);
195             if (validateBloodType(newnode->data.bloodtype)) {
196                 errorInput();
197             }
198         } while (!validateBloodType(newnode->data.bloodtype));
199         do {
200             cout << "Address: ";
201             getline(cin, newnode->data.address);
202             if (validateAddress(newnode->data.address)) {
203                 errorInput();
204             }
205         } while (!validateAddress(newnode->data.address));
206         do {
207             cout << "Neighborhood Association: ";
208             getline(cin, newnode->data.neighborhoodAssociation);
209             if (validateNeighborhoodAssociation(newnode->data.neighborhoodAssociation)) {
210                 errorInput();
211             }
212         } while (!validateNeighborhoodAssociation(newnode->data.neighborhoodAssociation));
213         do {
214             cout << "Community Association: ";
215             getline(cin, newnode->data.communityAssociation);
216             if (validateCommunityAssociation(newnode->data.communityAssociation)) {
217                 errorInput();
218             }
219         } while (!validateCommunityAssociation(newnode->data.communityAssociation));
220         do {
221             cout << "Sub District: ";
222             getline(cin, newnode->data.subDistrict);
223             if (validateSubDistrict(newnode->data.subDistrict)) {
224                 errorInput();
225             }
226         } while (!validateSubDistrict(newnode->data.subDistrict));
227         do {
228             cout << "District: ";
229             getline(cin, newnode->data.district);
230             if (validateDistrict(newnode->data.district)) {
231                 errorInput();
232             }
233         } while (!validateDistrict(newnode->data.district));
234         do {
235             cout << "Religion: ";
236             getline(cin, newnode->data.religion);
237             if (validateReligion(newnode->data.religion)) {
238                 errorInput();
239             }
240         } while (!validateReligion(newnode->data.religion));
241         do {
242             cout << "Married: ";
243             getline(cin, newnode->data.married);
244             if (validateMarried(newnode->data.married)) {
245                 errorInput();
246             }
247         } while (!validateMarried(newnode->data.married));
248         do {
249             cout << "Job: ";
250             getline(cin, newnode->data.job);
251             if (validateJob(newnode->data.job)) {
252                 errorInput();
253             }
254         } while (!validateJob(newnode->data.job));
255         do {
256             cout << "Created Date: ";
257             getline(cin, newnode->data.created.date);
258             if (validateDate(newnode->data.created.date)) {
259                 errorInput();
260             }
261         } while (!validateDate(newnode->data.created.date));
262         do {
263             cout << "Created Month: ";
264             getline(cin, newnode->data.created.month);
265             if (validateMonth(newnode->data.created.month)) {
266                 errorInput();
267             }
268         } while (!validateMonth(newnode->data.created.month));
269         do {
270             cout << "Created Year: ";
271             getline(cin, newnode->data.created.year);
272             if (validateYear(newnode->data.created.year)) {
273                 errorInput();
274             }
275         } while (!validateYear(newnode->data.created.year));
276         newnode->next = head;
277         head = newnode;
278         count++;
279     }
280
281     void display() {
282         Node* temp = head;
283         while (temp != nullptr) {
284             cout << "Province: " << temp->data.province << endl;
285             cout << "City: " << temp->data.city << endl;
286             cout << "Nia: " << temp->data.nia << endl;
287             cout << "Name: " << temp->data.name << endl;
288             cout << "Born Place: " << temp->data.bornPlace << endl;
289             cout << "Date: " << temp->data.born.date << endl;
290             cout << "Month: " << temp->data.born.month << endl;
291             cout << "Year: " << temp->data.born.year << endl;
292             cout << "Gender: " << temp->data.gender << endl;
293             cout << "Blood Type: " << temp->data.bloodtype << endl;
294             cout << "Address: " << temp->data.address << endl;
295             cout << "Neighborhood Association: " << temp->data.neighborhoodAssociation << endl;
296             cout << "Community Association: " << temp->data.communityAssociation << endl;
297             cout << "Sub District: " << temp->data.subDistrict << endl;
298             cout << "District: " << temp->data.district << endl;
299             cout << "Religion: " << temp->data.religion << endl;
300             cout << "Married: " << temp->data.married << endl;
301             cout << "Job: " << temp->data.job << endl;
302             cout << "Created Date: " << temp->data.created.date << endl;
303             cout << "Created Month: " << temp->data.created.month << endl;
304             cout << "Created Year: " << temp->data.created.year << endl;
305             temp = temp->next;
306         }
307     }
308
309     void deleteNode() {
310         Node* temp = head;
311         Node* prev = nullptr;
312         while (temp != nullptr) {
313             if (temp->next == nullptr) {
314                 delete temp;
315                 head = prev;
316                 count--;
317                 return;
318             }
319             prev = temp;
320             temp = temp->next;
321         }
32
```

```

152     do {
153         cout << "Nama : ";
154         getline(cin, newNode->data.name);
155         if (!validateName(newNode->data.name)) {
156             errorInput();
157         }
158     } while (!validateName(newNode->data.name));
159     do {
160         cout << "Tempat lahir : ";
161         getline(cin, newNode->data.born.place);
162         if (!validatePlace(newNode->data.born.place)) {
163             errorInput();
164         }
165     } while (!validatePlace(newNode->data.born.place));
166     do {
167         cout << "Tanggal lahir : ";
168         cin >> newNode->data.born.date;
169         if (!validateDate(newNode->data.born.date)) {
170             errorInput();
171         }
172     } while (!validateDate(newNode->data.born.date));
173     do {
174         cout << "Jalan lahir : ";
175         cin >> newNode->data.born.month;
176         if (!validateMonth(newNode->data.born.month)) {
177             errorInput();
178         }
179     } while (!validateMonth(newNode->data.born.month));
180     do {
181         cout << "Jalan lahir : ";
182         cin >> newNode->data.born.year;
183         if (!validateYear(newNode->data.born.year)) {
184             errorInput();
185         }
186     } while (!validateYear(newNode->data.born.year));
187     do {
188         cout << "jenis kelamin (laki-laki, perempuan) : ";
189         cin >> newNode->data.gender;
190         if (!validateGender(newNode->data.gender)) {
191             errorInput();
192         }
193     } while (!validateGender(newNode->data.gender));
194     cin.ignore();
195     do {
196         cout << "Golongan Darah : ";
197         getline(cin, newNode->data.bloodType);
198         if (!validateBloodType(newNode->data.bloodType)) {
199             errorInput();
200         }
201     } while (!validateBloodType(newNode->data.bloodType));
202     do {
203         cout << "Alamat : ";
204         getline(cin, newNode->data.address.address);
205         if (!validateAddress(newNode->data.address.address)) {
206             errorInput();
207         }
208     } while (!validateAddress(newNode->data.address.address));
209     do {
210         cout << "RT : ";
211         cin >> newNode->data.address.neighborhoodAssociation;
212         if (!validateNeighborhoodAssociation(newNode->data.address.neighborhoodAssociation)) {
213             errorInput();
214         }
215     } while (!validateNeighborhoodAssociation(newNode->data.address.neighborhoodAssociation));
216     do {
217         cout << "RW : ";
218         cin >> newNode->data.address.communityAssociation;
219         if (!validateCommunityAssociation(newNode->data.address.communityAssociation)) {
220             errorInput();
221         }
222     } while (!validateCommunityAssociation(newNode->data.address.communityAssociation));
223     cin.ignore();
224     do {
225         cout << "Kel/Desa : ";
226         getline(cin, newNode->data.address.subDistrict);
227         if (!validateSubDistrict(newNode->data.address.subDistrict)) {
228             errorInput();
229         }
230     } while (!validateSubDistrict(newNode->data.address.subDistrict));
231     do {
232         cout << "Kecamatan : ";
233         getline(cin, newNode->data.address.district);
234         if (!validateDistrict(newNode->data.address.district)) {
235             errorInput();
236         }
237     } while (!validateDistrict(newNode->data.address.district));
238     do {
239         cout << "Agama (Islam, Kristen, Katolik, Hindu, Buddha, Konghucu) : ";
240         getline(cin, newNode->data.religion);
241         if (!validateReligion(newNode->data.religion)) {
242             errorInput();
243         }
244     } while (!validateReligion(newNode->data.religion));
245     do {
246         cout << "Status Perkawinan (Kawin, Belum Kawin) : ";
247         getline(cin, newNode->data.married);
248         if (!validateMarried(newNode->data.married)) {
249             errorInput();
250         }
251     } while (!validateMarried(newNode->data.married));
252     do {
253         cout << "Pekerjaan : ";
254         getline(cin, newNode->data.job);
255         if (!validateJob(newNode->data.job)) {
256             errorInput();
257         }
258     } while (!validateJob(newNode->data.job));
259     newNode->data.citizenship = "WNI";
260     newNode->data.validUntil = "2024-12-31";
261     time_t now = time(0);
262     tm *ltm = localtime(&now);
263     newNode->data.created.date = ltm->tm_mday;
264     newNode->data.created.month = ltm->tm_mon + 1;
265     newNode->data.created.year = ltm->tm_year + 1900;
266     for (int i = 0; i < newNode->data.religion.length(); i++) {
267         newNode->data.religion[i] = toupper(newNode->data.religion[i]);
268     }
269     for (int i = 0; i < newNode->data.married.length(); i++) {
270         newNode->data.married[i] = toupper(newNode->data.married[i]);
271     }
272     ofstream file;
273     file.open("data.txt", ios::app);
274     file << "Provinsi : " << newNode->data.province << endl;
275     file << "Kota/Kabupaten : " << newNode->data.city << endl;
276     file << "RT : " << newNode->data.rtk << endl;
277     file << "RW : " << newNode->data.rtw << endl;
278     file << "Tempat/Tgl lahir : " << newNode->data.born.place << ", " << setfill('0') << setw(2) << newNode->data.born.date << "-" << setfill('0') << setw(2) << newNode->data.born.month << "-" << newNode->data.born.year << endl;
279     file << "Jenis Kelamin : " << newNode->data.gender << ", " << setfill('0') << setw(2) << newNode->data.born.date << "-" << setfill('0') << setw(2) << newNode->data.born.month << "-" << newNode->data.born.year << endl;
280     file << "Gol.Darah : " << newNode->data.bloodType << endl;
281     file << "Alamat : " << newNode->data.address.address << endl;
282     file << "RT/RW : " << newNode->data.address.neighborhoodAssociation << "/" << setfill('0') << setw(2) << newNode->data.address.communityAssociation << endl;
283     file << "Kel/Desa : " << newNode->data.address.subDistrict << endl;
284     file << "Kecamatan : " << newNode->data.address.district << endl;
285     file << "Agama : " << newNode->data.religion << endl;
286     file << "Status Perkawinan : " << newNode->data.married << endl;
287     file << "Pekerjaan : " << newNode->data.job << endl;
288     file << "Kewarganegaraan : " << newNode->data.citizenship << endl;
289     file << "Validasi hingga : " << newNode->data.validUntil << endl;
290     file << "Tanggal : " << setfill('0') << setw(2) << newNode->data.created.date << "-" << setfill('0') << setw(2) << newNode->data.created.month << "-" << newNode->data.created.year << endl;
291     file << "===== " << endl;
292     file.close();
293     newNode->reset();
294     head = newNode;
295     count++;
296 }

```

```

902 void output() {
903     cout << "=====KARTU TANPA PERUBAHAN" << endl;
904     cout << "=====KARTU TANPA PERUBAHAN" << endl;
905     Node* current = head;
906     while (current != nullptr) {
907         int provincelength = current->data.province.length();
908         int citylength = current->data.city.length();
909         int maxlength = provincelength > citylength ? provincelength : citylength;
910         for (int j = 0; j < (maxlength - provincelength) / 2; j++) {
911             cout << " ";
912         }
913         cout << " " << current->data.province << endl;
914         for (int j = 0; j < (maxlength - citylength) / 2; j++) {
915             cout << " ";
916         }
917         cout << " " << current->data.city << endl;
918         cout << "NIK " << current->data.nik << endl;
919         cout << "Nama " << current->data.name << endl;
920         cout << "Tempat/tgl lahir " << current->data.horn.place << ", " << setfill('0') << setw(2) << current->data.horn.date << " " << setfill('0') << setw(2) << current->data.horn.month << " " << current->data.horn.year << endl;
921         cout << "Golongan darah " << current->data.bloodtype << endl;
922         cout << "No/da " << setfill('0') << setw(9) << current->data.address.neighborhoodassociation << " " << setfill('0') << setw(3) << current->data.address.communityassociation << endl;
923         cout << "No/da " << current->data.address.subdistrict << endl;
924         cout << "No/da " << current->data.address.district << endl;
925         cout << "Agama " << current->data.religion << endl;
926         cout << "Status Perkawinan " << current->data.marital << endl;
927         cout << "setfill(' ') << setw(4) << "foto.jpg" << endl;
928         cout << "Pekerjaan " << current->data.job << endl;
929         cout << "No/da " << current->data.validuntil << endl;
930         cout << "setfill(' ') << setw(4) << to_string(current->data.created.date).length() << " " << setfill('0') << setw(2) << current->data.created.date << " " << setfill('0') << setw(2) << current->data.created.month << " " << current->data.created.year << endl;
931         cout << "setfill(' ') << setw(4) << "TID" << endl;
932         cout << "=====KARTU TANPA PERUBAHAN" << endl;
933         current = current->next;
934     }
935 }
936 void initialize() {
937     ifstream file;
938     file.open("data.txt");
939     string line;
940     count = 0;
941     while (getline(file, line)) {
942         if (line.find("NIK") != string::npos) {
943             count++;
944         }
945     }
946     cout << "banyak NIK: " << count << endl;
947     file.close();
948     this->head = nullptr;
949     Node* tail = nullptr;
950     for (int i = 0; i < count; i++) {
951         Data readdata;
952         while (getline(file, line)) {
953             if (line.find("Province") != string::npos) readdata.province = line.substr(line.find(" ") + 2);
954             else if (line.find("kota/kabupaten") != string::npos) readdata.city = line.substr(line.find(" ") + 2);
955             else if (line.find("NIK") != string::npos) readdata.nik = line.substr(line.find(" ") + 2);
956             else if (line.find("Nama") != string::npos) readdata.name = line.substr(line.find(" ") + 2);
957             else if (line.find("Tempat/tgl lahir") != string::npos) {
958                 readdata.horn.place = line.substr(line.find(" ") + 2);
959                 readdata.horn.date = stoi(line.substr(line.find(" ") + 2, 2));
960                 readdata.horn.month = stoi(line.substr(line.find(" ") + 2, 2));
961                 readdata.horn.year = stoi(line.substr(line.find(" ") + 2, 4));
962             }
963             else if (line.find("Golongan darah") != string::npos) {
964                 readdata.bloodtype = line.substr(line.find(" ") + 2);
965             }
966             else if (line.find("No/da") != string::npos) {
967                 readdata.address.neighborhoodassociation = stoi(line.substr(line.find(" ") + 2, 9));
968                 readdata.address.communityassociation = stoi(line.substr(line.find(" ") + 2, 3));
969             }
970             else if (line.find("No/da") != string::npos) {
971                 readdata.address.subdistrict = line.substr(line.find(" ") + 2);
972             }
973             else if (line.find("No/da") != string::npos) {
974                 readdata.address.district = line.substr(line.find(" ") + 2);
975             }
976             else if (line.find("Agama") != string::npos) {
977                 readdata.religion = line.substr(line.find(" ") + 2);
978             }
979             else if (line.find("Status Perkawinan") != string::npos) {
980                 readdata.marital = line.substr(line.find(" ") + 2);
981             }
982             else if (line.find("Pekerjaan") != string::npos) {
983                 readdata.job = line.substr(line.find(" ") + 2);
984             }
985             else if (line.find("No/da") != string::npos) {
986                 readdata.validuntil = line.substr(line.find(" ") + 2);
987             }
988             else if (line.find("Tanggal") != string::npos) {
989                 readdata.created.date = stoi(line.substr(line.find(" ") + 2, 2));
990                 readdata.created.month = stoi(line.substr(line.find(" ") + 2, 2));
991                 readdata.created.year = stoi(line.substr(line.find(" ") + 2, 4));
992             }
993             else if (line.find("=====KARTU TANPA PERUBAHAN") != string::npos) {
994                 Node* newnode = new Node(readdata, nullptr);
995                 if (head == nullptr) {
996                     head = newnode;
997                     tail = newnode;
998                 }
999                 else {
1000                     tail->next = newnode;
1001                     tail = newnode;
1002                     break;
1003                 }
1004             }
1005         }
1006         file.close();
1007     }
1008     void deleteHead() {
1009         if (head == nullptr) {
1010             cout << "Data kosong" << endl;
1011             return;
1012         }
1013         Node* temp = head;
1014         head = head->next;
1015         delete temp;
1016         cout << "Data berhasil dihapus" << endl;
1017         count--;
1018     }
1019     ifstream file;
1020     file.open("data.txt");
1021     ofstream tempfile;
1022     tempfile.open("temp.txt");
1023     string line;
1024     int linecount = 0;
1025     while (getline(file, line)) {
1026         if (line.find("NIK") != string::npos) {
1027             linecount++;
1028         }
1029         if (linecount == 1) {
1030             tempfile << line << endl;
1031         }
1032     }
1033     file.close();
1034     tempfile.close();
1035     remove("data.txt");
1036     rename("temp.txt", "data.txt");
1037     void deleteTail() {
1038         if (head == nullptr) {
1039             cout << "Linked list kosong" << endl;
1040             return;
1041         }
1042         Node* current = head;
1043         Node* prev = nullptr;
1044         while (current->next != nullptr) {
1045             prev = current;
1046             current = current->next;
1047         }
1048         if (prev == nullptr) {
1049             head = nullptr;
1050         }
1051         else {
1052             prev->next = nullptr;
1053         }
1054         delete current;
1055         cout << "Data berhasil dihapus" << endl;
1056         count--;
1057     }
1058     ifstream file;
1059     file.open("data.txt");
1060     ofstream tempfile;
1061     tempfile.open("temp.txt");
1062     string line;
1063     int linecount = 0;
1064     while (getline(file, line)) {
1065         if (line.find("NIK") != string::npos) {
1066             linecount++;
1067         }
1068     }

```

```

472     }
473     if (lineCount != count + 1) {
474         tempFile << line << endl;
475     }
476 }
477 file.close();
478 tempFile.close();
479 remove("data.txt");
480 remove("temp.txt", "data.txt");
481 }
482 }
483
484 int main() {
485     Ktp Ktp;
486     int choice;
487     while(true){
488         Ktp.initialize();
489         cout << "Menu: " << endl;
490         cout << "1. Input Data" << endl;
491         cout << "2. Tampilkan Data" << endl;
492         cout << "3. Delete Data" << endl;
493         cout << "4. Keluar" << endl;
494         cout << "Pilihan: ";
495         cin >> choice;
496         system("cls");
497         cin.ignore();
498         switch (choice) {
499             case 1:
500                 Ktp.input();
501                 cout << "Data berhasil diinput" << endl;
502                 break;
503             case 2:
504                 Ktp.output();
505                 break;
506             case 3:
507                 while(true){
508                     system("cls");
509                     cout << "Menu: " << endl;
510                     cout << "1. Delete data di depan" << endl;
511                     cout << "2. Delete data di belakang" << endl;
512                     cout << "3. Batal" << endl;
513                     cout << "Pilihan: ";
514                     cin >> choice;
515                     switch(choice){
516                         case 1:
517                             Ktp.deleteHead();
518                             break;
519                         case 2:
520                             Ktp.deleteTail();
521                             break;
522                         case 3:
523                             break;
524                         default:
525                             cout << "Pilihan tidak valid" << endl;
526                             system("pause");
527                             break;
528                     }
529                 }
530             case 4:
531                 break;
532             default:
533                 return EXIT_SUCCESS;
534             default:
535                 cout << "Pilihan tidak valid" << endl;
536                 break;
537             }
538         }
539         system("pause");
540         system("cls");
541     }
542 }

```

Gunakan struktur dari postes pertemuan 3 (data KTP) sebagai data link list. Buat operasi tambah dan hapus data

a. no ganjil : di belakang

b. no genap di depan

Program jalan (Screenshot input output)

```

Banyak KTP: 12
Menu:
1. Input Data
2. Tampilkan Data
3. Delete Data
4. Keluar
Pilihan: 

```

Tampilan awal Program

Mencoba input data, karena merupakan input data dari belakang (insertAtTail), oleh karena itu akan tersimpan di data.txt pada bagian bawah

```

164 .....}
165 .....} while (!validateBornPlace(newNode->data.born.place));
166 .....}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Alamat : PERUMAHAN PANDAWA GARDEN
RT/RW : 000/000
Kel/Desa : KUTUH
Kecamatan : KUTA SELATAN
Agama : ISLAM
Status Perkawinan : KAWIN Foto.jpg
Pekerjaan : OTHERS
Kewarganegaraan : MNI 17-10-2023
Berlaku Hingga : SELUMUR HIDUP TTD

PROVINSI JAWA TENGAH
KABUPATEN PEKALONGAN

NIK : 3329091003700012
Nama : AKAAT FAOZI
Tempat/Tgl Lahir : PEKALONGAN, 10-03-1978
Jenis Kelamin : PEREMPUAN Gol.Darah : A
Alamat : DUSUN KALMAN
RT/RW : 002/005
Kel/Desa : KESESI
Kecamatan : KESESI
Agama : ISLAM
Status Perkawinan : KAWIN Foto.jpg
Pekerjaan : Pedagang
Kewarganegaraan : MNI 17-10-2023
Berlaku Hingga : SELUMUR 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 : BELLUM KAWIN Foto.jpg
Pekerjaan : PETANI/PEKEBUN
Kewarganegaraan : MNI 17-10-2023
Berlaku Hingga : SELUMUR HIDUP TTD

PROVINSI JAWA TENGAH
KOTA SURABAYA

NIK : 3372011210770001
Nama : SELAMAT IDUL ADHA
Tempat/Tgl Lahir : SURABAYA, 12-10-1977
Jenis Kelamin : PEREMPUAN Gol.Darah : AB
Alamat : BARON CILIK
RT/RW : 004/005
Kel/Desa : BUMI
Kecamatan : LAMEYAN
Agama : ISLAM
Status Perkawinan : KAWIN Foto.jpg
Pekerjaan : PERDAGANGAN
Kewarganegaraan : MNI 17-10-2023
Berlaku Hingga : SELUMUR HIDUP TTD

PROVINSI SULAWESI SELATAN
KABUPATEN SOPPENG

NIK : 7312042510720002
Nama : ABDURRAUF, S.Pd, M.Pd
Tempat/Tgl Lahir : CELLENGENGE, 25-10-1972
Jenis Kelamin : PEREMPUAN Gol.Darah : O
Alamat : JL. MERDEKA NO.43
RT/RW : 001/004
Kel/Desa : BILA
Kecamatan : LALABATA
Agama : ISLAM
Status Perkawinan : KAWIN Foto.jpg
Pekerjaan : PEGAWAI NEGERI SIPIL
Kewarganegaraan : MNI 17-10-2023
Berlaku Hingga : SELUMUR HIDUP TTD

Press any key to continue . . .

Input data sebelum ditambahkan

Mencoba input data yang baru



Berikut input data merupakan

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

Provinsi: PROVINSI KEPULAUAN RIAU
Kota: KABUPATEN KEPULAUAN ANAMBAS
NIK: 2105042604950001
Nama: BENI AFRIANTO
Tempat Lahir: MENGKAIT
Tanggal Lahir: 26
Bulan Lahir: 4
Tahun Lahir: 19995
Input tidak valid
Tahun Lahir:
1995
Jenis kelamin (0=laki-laki, 1=perempuan): 0
Golongan Darah: 0
Alamat: JL. BATU ABLAI
RT: 1
RW: 1
Kel/Desa: KELURAHAN TEREMPA
Kecamatan: SIANTAN
Agama (Islam, Kristen, Katolik, Hindu, Buddha, Konghucu): Katolik
Status Perkawinan (Kawin, Belum Kawin): Belum Kawin
Pekerjaan: PELAJAR/MAHASISWA
Data berhasil diinput
Press any key to continue . . .

```

Berikut mengecek hasil input pada menu ke-2 yaitu Tampilkan Data

Operasi menambah data

```

NIK          : 7312042510720002
Nama         : ABDURRAUF, S.Pd, M.Pd
Tempat/Tgl Lahir : CELLENGENGE, 25-10-1972
Jenis Kelamin  : PEREMPUAN           Gol.Darah : 0
Alamat        : JL. MERDEKA NO.43
RT/RW         : 001/004
Kel/Desa      : BILA
Kecamatan     : LALABATA
Agama         : ISLAM
Status Perkawinan : KAWIN           Foto.jpg
Pekerjaan     : PEGAWAI NEGERI SIPIL
Kewarganegaraan : WNI              17-10-2023
Berlaku Hingga : SEUMUR HIDUP       TTD
=====
PROVINSI KEPULAUAN RIAU
KABUPATEN KEPULAUAN ANAMBAS
NIK          : 2105042604950001
Nama         : BENI AFRIANTO
Tempat/Tgl Lahir : MENGKAIT, 26-04-1995
Jenis Kelamin  : PEREMPUAN           Gol.Darah : 0
Alamat        : JL. BATU ABLAI
RT/RW         : 001/001
Kel/Desa      : KELURAHAN TEREMPA
Kecamatan     : SIANTAN
Agama         : KATOLIK
Status Perkawinan : BELUM KAWIN       Foto.jpg
Pekerjaan     : PELAJAR/MAHASISWA
Kewarganegaraan : WNI              24-10-2023
Berlaku Hingga : SEUMUR HIDUP       TTD
=====
Press any key to continue . . .

```

Nah, akan muncul pada akhir data karena merupakan menambahkan data di belakang (insertAtTail)

Berikut akan mencoba menghapus 2x. Kita bisa menggunakan menu ke-3 yaitu Delete Data

```

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

```

Karena saya **NIM GANJI**, akan mencoba delete data di belakang 2x yaitu mencoba menu ke-2.


```

Pekerjaan      : Pedagang
Kewarganegaraan : WNI
Berlaku Hingga : SEUMUR HIDUP
17-10-2023
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
Pekerjaan     : PETANI/PEKEBUN
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
Pekerjaan     : PERDAGANGAN
Kewarganegaraan : WNI
Berlaku Hingga : SEUMUR HIDUP
17-10-2023
TTD
=====
Press any key to continue

```

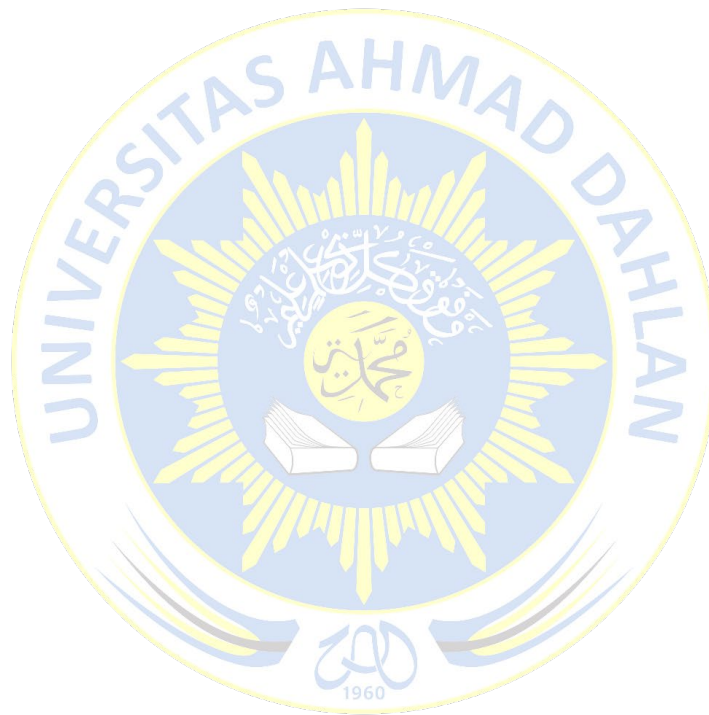
Ternyata 2 data berhasil di hapus, data yang masih ada selanjutnya akan disimpan kembali di data.txt

Apabila ingin menghapus data pada dari depan (delAtHead)

Data sebelum

KARTU TANDA PENDUDUK			
<div> <div>PROVINSI KALIMANTAN TENGAH</div> <div>KABUPATEN KOTAWARINGIN BARAT</div> </div>			
NIK	: 6201052907040001		
Nama	: RENDIE ABDI SAPUTRA		
Tempat/Tgl Lahir	: PANGKALAN BUN, 29-07-2004		
Jenis Kelamin	: PEREMPUAN	Gol.Darah : B	
Alamat	: JL. A. YANI		
RT/RW	: 006/003		
Kel/Desa	: SUMBER AGUNG		
Kecamatan	: PANGKALAN LADA		
Agama	: ISLAM		
Status Perkawinan	: BELUM KAWIN	Foto.jpg	
Pekerjaan	: PELAJAR/MAHASISWA		
Kewarganegaraan	: WNI	02-08-2021	
Berlaku Hingga	: SEUMUR HIDUP	TTD	
<div> <div>PROVINSI KALIMANTAN TENGAH</div> <div>KABUPATEN KOTAWARINGIN BARAT</div> </div>			
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	
<div> <div>PROVINSI DAERAH ISTIMEWA YOGYAKARTA</div> <div>KOTA YOGYAKARTA</div> </div>			
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	
<div> <div>PROVINSI DAERAH ISTIMEWA YOGYAKARTA</div> <div>KOTA YOGYAKARTA</div> </div>			
NIK	: 3203012503770011		

Sesudah, maka data KTP yang di depan atau head akan dihapus, yaitu akan menghapus data KTP Rendie.



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	: JL. SELAMET PERUMAHAN RANCABALI NO.40		
RT/RW	: 002/004		
Kel/Desa	: MUKA		
Kecamatan	: CIANDUR		
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		

Operasi penambahan data

Sudah jelas terlihat di prosedur input() pada class KTP.

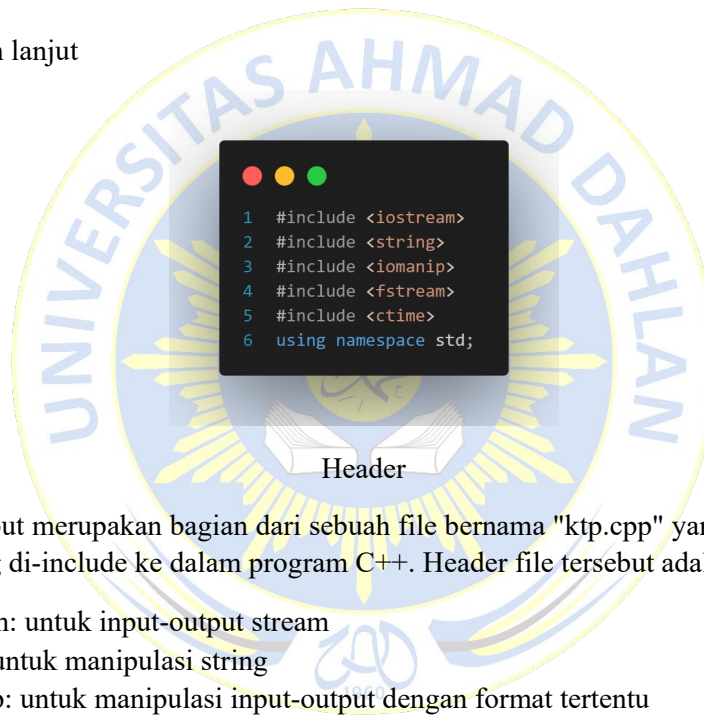
Operasi menghapus data

Sudah jelas terlihat di prosedur delAtTail() pada class KTP. (menghapus dari belakang)

Class

Class yang digunakan dalam program ini adalah class KTP.

Penjelasan lebih lanjut



Kode tersebut merupakan bagian dari sebuah file bernama "ktp.cpp" yang berisi beberapa header file yang di-include ke dalam program C++. Header file tersebut adalah:

- iostream: untuk input-output stream
- string: untuk manipulasi string
- iomanip: untuk manipulasi input-output dengan format tertentu
- fstream: untuk manipulasi file
- ctime: untuk manipulasi waktu

Kode "using namespace std;" digunakan untuk menghindari penulisan "std::" sebelum setiap penggunaan fungsi atau objek dari namespace std. Karena ingin lebih memudahkan penulisan, daripada pertemuan sebelumnya, oleh karena itu saya menambahkan kode ini.



```
8 void errorInput(){  
9     cout << "Input tidak valid" << endl;  
10 }  
11
```

Menangani error input (tidak valid)

Fungsi `errorInput()` adalah sebuah fungsi dalam bahasa pemrograman C++ yang bertujuan untuk menampilkan pesan "Input tidak valid" ke dalam output console. Fungsi ini tidak menerima parameter apapun dan hanya menampilkan pesan tersebut ke dalam output console. Fungsi ini dapat dipanggil dari bagian manapun dalam program C++ untuk memberikan pesan error jika input yang diberikan tidak valid.



```

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

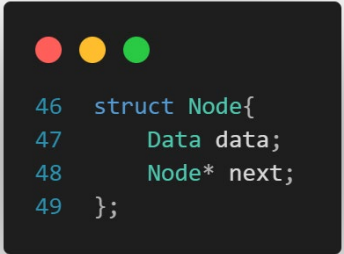
```

Struct Data (KTP)

GitHub Copilot: Struktur `Data` adalah sebuah struktur dalam bahasa pemrograman C++ yang memiliki beberapa variabel anggota (member variables) yang merepresentasikan data pada sebuah Kartu Tanda Penduduk (KTP). Struktur ini memiliki variabel anggota sebagai berikut:

- `province` (tipe data: string): merepresentasikan provinsi tempat tinggal pemilik KTP.
- `city` (tipe data: string): merepresentasikan kota tempat tinggal pemilik KTP.

- `nik` (tipe data: string): merepresentasikan Nomor Induk Kependudukan (NIK) pemilik KTP.
- `name` (tipe data: string): merepresentasikan nama lengkap pemilik KTP.
- `born` (tipe data: struct Born): merupakan sebuah struktur yang memiliki variabel anggota sebagai berikut:
 - `place` (tipe data: string): merepresentasikan tempat lahir pemilik KTP.
 - `date` (tipe data: int): merepresentasikan tanggal lahir pemilik KTP.
 - `month` (tipe data: int): merepresentasikan bulan lahir pemilik KTP.
 - `year` (tipe data: int): merepresentasikan tahun lahir pemilik KTP.
 - `gender` (tipe data: bool): merepresentasikan jenis kelamin pemilik KTP.
 - `bloodType` (tipe data: string): merepresentasikan golongan darah pemilik KTP.
- `address` (tipe data: struct Address): merupakan sebuah struktur yang memiliki variabel anggota sebagai berikut:
 - o `address` (tipe data: string): merepresentasikan alamat lengkap pemilik KTP.
 - o `neighborhoodAssociation` (tipe data: int): merepresentasikan nomor RT (Rukun Tetangga) pemilik KTP.
 - o `communityAssociation` (tipe data: int): merepresentasikan nomor RW (Rukun Warga) pemilik KTP.
 - o `district` (tipe data: string): merepresentasikan kelurahan tempat tinggal pemilik KTP.
 - o `subDistrict` (tipe data: string): merepresentasikan kecamatan tempat tinggal pemilik KTP.
- `religion` (tipe data: string): merepresentasikan agama pemilik KTP.
- `isMaried` (tipe data: string): merepresentasikan status pernikahan pemilik KTP.
- `job` (tipe data: string): merepresentasikan pekerjaan pemilik KTP.
- `citizenship` (tipe data: string): merepresentasikan kewarganegaraan pemilik KTP.
- `validUntil` (tipe data: string): merepresentasikan tanggal berakhirnya masa berlaku KTP.
- `created` (tipe data: struct Created): merupakan sebuah struktur yang memiliki variabel anggota sebagai berikut:
 - o `date` (tipe data: int): merepresentasikan tanggal pembuatan KTP.
 - o `month` (tipe data: int): merepresentasikan bulan pembuatan KTP.
 - o `year` (tipe data: int): merepresentasikan tahun pembuatan KTP.



```

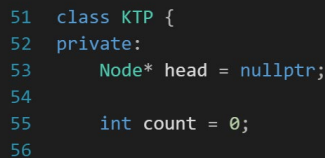
46 struct Node{
47     Data data;
48     Node* next;
49 };

```

Struct Node

Struktur Node adalah sebuah struktur dalam bahasa pemrograman C++ yang merepresentasikan sebuah simpul (node) pada sebuah linked list. Struktur ini memiliki dua variabel anggota (member variables) yaitu:

- data (tipe data: struct Data): merepresentasikan data yang disimpan pada simpul tersebut. Variabel ini bertipe data struct Data yang merupakan sebuah struktur yang memiliki beberapa variabel anggota yang merepresentasikan data pada sebuah Kartu Tanda Penduduk (KTP).
- next (tipe data: Node*): merepresentasikan alamat dari simpul berikutnya pada linked list. Variabel ini bertipe data Node* yang merupakan sebuah pointer yang menunjuk ke simpul berikutnya pada linked list. Jika simpul tersebut merupakan simpul terakhir pada linked list, maka nilai variabel next akan bernilai NULL.



```

51 class KTP {
52 private:
53     Node* head = nullptr;
54
55     int count = 0;
56

```

Kelas `KTP` adalah sebuah kelas dalam bahasa pemrograman C++ yang merepresentasikan sebuah linked list yang berisi data-data pada sebuah Kartu Tanda Penduduk (KTP). Kelas ini memiliki dua variabel anggota (member variables) yang bersifat private yaitu:

- `head` (tipe data: Node*): merepresentasikan alamat dari simpul pertama pada linked list. Variabel ini bertipe data `Node*` yang merupakan sebuah pointer yang menunjuk ke simpul pertama pada linked list. Jika linked list kosong, maka nilai variabel `head` akan bernilai `NULL`.

- `count` (tipe data: int): merepresentasikan jumlah simpul pada linked list. Variabel ini bertipe data `int` dan diinisialisasi dengan nilai `0` pada saat pembuatan objek dari kelas `KTP`.

Kedua variabel anggota tersebut bersifat private sehingga hanya dapat diakses dari dalam kelas `KTP` dan tidak dapat diakses dari luar kelas.

```

58 bool validateProvince(string province) {
59     return province.length() <= 50;
60 }
61
62
63 bool validateCity(string city) {
64     return city.length() <= 50;
65 }
66
67
68 bool validateNik(string nik) {
69     return nik.length() == 16;
70 }
71
72 bool validateName(string name) {
73     return name.length() <= 50;
74 }
75
76 bool validateBornPlace(string bornPlace) {
77     return bornPlace.length() <= 20;
78 }
79
80 bool validateDate(int date) {
81     return date >= 1 && date <= 31;
82 }
83
84 bool validateMonth(int month) {
85     return month >= 1 && month <= 12;
86 }
87
88 bool validateYear(int year) {
89     return year >= 1900 && year <= 2023;
90 }
91
92 bool validateGender(int gender) {
93     return gender == 0 || gender == 1;
94 }
95
96 bool validateBloodType(string bloodType) {
97     return bloodType == "A" || bloodType == "B" || bloodType == "AB" || bloodType == "O";
98 }
99
100 bool validateAddress(string address) {
101     return address.length() <= 100;
102 }
103
104
105 bool validateNeighborhoodAssociation(int neighborhoodAssociation) {
106     return neighborhoodAssociation >= 0 && neighborhoodAssociation <= 999;
107 }
108
109 bool validateCommunityAssociation(int communityAssociation) {
110     return communityAssociation >= 0 && communityAssociation <= 999;
111 }
112
113 bool validateSubDistrict(string subDistrict) {
114     return subDistrict.length() <= 20;
115 }
116
117 bool validateDistrict(string district) {
118     return district.length() <= 20;
119 }
120
121 bool validateReligion(string religion) {
122     return religion == "Islam" || religion == "Kristen" || religion == "Katolik" || religion == "Hindu" || religion == "Buddha" || religion == "Konghucu";
123 }
124
125 bool validateIsMarried(string isMarried) {
126     return isMarried == "Kawin" || isMarried == "Belum Kawin";
127 }
128
129 bool validateJob(string job) {
130     return job.length() <= 30;
131 }
132

```

Validasi input user

Fungsi-fungsi di atas adalah fungsi validasi dalam bahasa pemrograman C++ yang bertujuan untuk memeriksa apakah data yang dimasukkan sesuai dengan format yang diinginkan.

Fungsi-fungsi tersebut memiliki parameter masukan yang berbeda-beda tergantung pada jenis data yang akan divalidasi. Berikut adalah penjelasan dari masing-masing fungsi validasi:

- `validateProvince(string province)`: memeriksa apakah provinsi yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 50.
- `validateCity(string city)`: memeriksa apakah kota yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 50.
- `validateNik(string nik)`: memeriksa apakah Nomor Induk Kependudukan (NIK) yang dimasukkan memiliki panjang karakter sama dengan 16.
- `validateName(string name)`: memeriksa apakah nama yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 50.
- `validateBornPlace(string bornPlace)`: memeriksa apakah tempat lahir yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 20.
- `validateDate(int date)`: memeriksa apakah tanggal yang dimasukkan berada dalam rentang 1 hingga 31.
- `validateMonth(int month)`: memeriksa apakah bulan yang dimasukkan berada dalam rentang 1 hingga 12.
- `validateYear(int year)`: memeriksa apakah tahun yang dimasukkan berada dalam rentang 1900 hingga 2023.
- `validateGender(int gender)`: memeriksa apakah jenis kelamin yang dimasukkan bernilai 0 atau 1.
- `validateBloodType(string bloodType)`: memeriksa apakah golongan darah yang dimasukkan merupakan salah satu dari "A", "B", "AB", atau "O".
- `validateAddress(string address)`: memeriksa apakah alamat yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 100.
- `validateNeighborhoodAssociation(int neighborhoodAssociation)`: memeriksa apakah nomor RT (Rukun Tetangga) yang dimasukkan berada dalam rentang 0 hingga 999.
- `validateCommunityAssociation(int communityAssociation)`: memeriksa apakah nomor RW (Rukun Warga) yang dimasukkan berada dalam rentang 0 hingga 999.
- `validateSubDistrict(string subDistrict)`: memeriksa apakah kecamatan yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 20.
- `validateDistrict(string district)`: memeriksa apakah kelurahan yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 20.
- `validateReligion(string religion)`: memeriksa apakah agama yang dimasukkan merupakan salah satu dari "Islam", "Kristen", "Katolik", "Hindu", "Buddha", atau "Konghucu".
- `validateIsMarried(string isMarried)`: memeriksa apakah status pernikahan yang dimasukkan merupakan salah satu dari "Kawin" atau "Belum Kawin".
- `validateJob(string job)`: memeriksa apakah pekerjaan yang dimasukkan memiliki panjang karakter kurang dari atau sama dengan 30.

- ```

140 void Input() {
141 cout << "Name: ";
142 do {
143 cout << "Please enter a name: ";
144 getline(is, name, data.previos);
145 if (validatePrevious(name, data.previos)) {
146 errorInput();
147 }
148 } while (!validatePrevious(name, data.previos));
149 do {
150 cout << "City: ";
151 getline(is, name, data.city);
152 if (validateCity(name, data.city)) {
153 errorInput();
154 }
155 } while (!validateCity(name, data.city));
156 do {
157 cout << "NBI: ";
158 getline(is, name, data.nbi);
159 if (validateNBI(name, data.nbi)) {
160 errorInput();
161 }
162 } while (!validateNBI(name, data.nbi));
163 do {
164 cout << "Name: ";
165 getline(is, name, data.name);
166 if (validateName(name, data.name)) {
167 errorInput();
168 }
169 } while (!validateName(name, data.name));
170 do {
171 cout << "Country: ";
172 getline(is, name, data.born_place);
173 if (validateCountry(name, data.born_place)) {
174 errorInput();
175 }
176 } while (!validateCountry(name, data.born_place));
177 do {
178 cout << "Occupation: ";
179 cin << name;
180 if (validateOccupation(name, data.born_place)) {
181 errorInput();
182 }
183 } while (!validateOccupation(name, data.born_place));
184 do {
185 cout << "Month: ";
186 cin << name;
187 if (validateMonth(name, data.born_date)) {
188 errorInput();
189 }
190 } while (!validateMonth(name, data.born_date));
191 do {
192 cout << "Year: ";
193 cin << name;
194 if (validateYear(name, data.born_date)) {
195 errorInput();
196 }
197 } while (!validateYear(name, data.born_date));
198 do {
199 cout << "Gender: ";
200 cin << name;
201 if (validateGender(name, data.born_date)) {
202 errorInput();
203 }
204 } while (!validateGender(name, data.born_date));
205 do {
206 cout << "Hobby: ";
207 cin << name;
208 if (validateHobby(name, data.born_date)) {
209 errorInput();
210 }
211 } while (!validateHobby(name, data.born_date));
212 do {
213 cout << "Religion: ";
214 cin << name;
215 if (validateReligion(name, data.born_date)) {
216 errorInput();
217 }
218 } while (!validateReligion(name, data.born_date));
219 do {
220 cout << "Address: ";
221 cin << name;
222 if (validateAddress(name, data.born_date)) {
223 errorInput();
224 }
225 } while (!validateAddress(name, data.born_date));
226 do {
227 cout << "Phone: ";
228 cin << name;
229 if (validatePhone(name, data.born_date)) {
230 errorInput();
231 }
232 } while (!validatePhone(name, data.born_date));
233 do {
234 cout << "Email: ";
235 cin << name;
236 if (validateEmail(name, data.born_date)) {
237 errorInput();
238 }
239 } while (!validateEmail(name, data.born_date));
240 do {
241 cout << "Social Media: ";
242 cin << name;
243 if (validateSocialMedia(name, data.born_date)) {
244 errorInput();
245 }
246 } while (!validateSocialMedia(name, data.born_date));
247 do {
248 cout << "Education: ";
249 cin << name;
250 if (validateEducation(name, data.born_date)) {
251 errorInput();
252 }
253 } while (!validateEducation(name, data.born_date));
254 do {
255 cout << "Work: ";
256 cin << name;
257 if (validateWork(name, data.born_date)) {
258 errorInput();
259 }
260 } while (!validateWork(name, data.born_date));
261 do {
262 cout << "Marital Status: ";
263 cin << name;
264 if (validateMaritalStatus(name, data.born_date)) {
265 errorInput();
266 }
267 } while (!validateMaritalStatus(name, data.born_date));
268 do {
269 cout << "Spouse: ";
270 cin << name;
271 if (validateSpouse(name, data.born_date)) {
272 errorInput();
273 }
274 } while (!validateSpouse(name, data.born_date));
275 do {
276 cout << "Children: ";
277 cin << name;
278 if (validateChildren(name, data.born_date)) {
279 errorInput();
280 }
281 } while (!validateChildren(name, data.born_date));
282 do {
283 cout << "Pets: ";
284 cin << name;
285 if (validatePets(name, data.born_date)) {
286 errorInput();
287 }
288 } while (!validatePets(name, data.born_date));
289 do {
290 cout << "Hobbies: ";
291 cin << name;
292 if (validateHobbies(name, data.born_date)) {
293 errorInput();
294 }
295 } while (!validateHobbies(name, data.born_date));
296 do {
297 cout << "Interests: ";
298 cin << name;
299 if (validateInterests(name, data.born_date)) {
300 errorInput();
301 }
302 } while (!validateInterests(name, data.born_date));
303 do {
304 cout << "Skills: ";
305 cin << name;
306 if (validateSkills(name, data.born_date)) {
307 errorInput();
308 }
309 } while (!validateSkills(name, data.born_date));
310 do {
311 cout << "Languages: ";
312 cin << name;
313 if (validateLanguages(name, data.born_date)) {
314 errorInput();
315 }
316 } while (!validateLanguages(name, data.born_date));
317 do {
318 cout << "Travel: ";
319 cin << name;
320 if (validateTravel(name, data.born_date)) {
321 errorInput();
322 }
323 } while (!validateTravel(name, data.born_date));
324 do {
325 cout << "Food: ";
326 cin << name;
327 if (validateFood(name, data.born_date)) {
328 errorInput();
329 }
330 } while (!validateFood(name, data.born_date));
331 do {
332 cout << "Drinks: ";
333 cin << name;
334 if (validateDrinks(name, data.born_date)) {
335 errorInput();
336 }
337 } while (!validateDrinks(name, data.born_date));
338 do {
339 cout << "Clothing: ";
340 cin << name;
341 if (validateClothing(name, data.born_date)) {
342 errorInput();
343 }
344 } while (!validateClothing(name, data.born_date));
345 do {
346 cout << "Accessories: ";
347 cin << name;
348 if (validateAccessories(name, data.born_date)) {
349 errorInput();
350 }
351 } while (!validateAccessories(name, data.born_date));
352 do {
353 cout << "Grooming: ";
354 cin << name;
355 if (validateGrooming(name, data.born_date)) {
356 errorInput();
357 }
358 } while (!validateGrooming(name, data.born_date));
359 do {
360 cout << "Beauty: ";
361 cin << name;
362 if (validateBeauty(name, data.born_date)) {
363 errorInput();
364 }
365 } while (!validateBeauty(name, data.born_date));
366 do {
367 cout << "Fitness: ";
368 cin << name;
369 if (validateFitness(name, data.born_date)) {
370 errorInput();
371 }
372 } while (!validateFitness(name, data.born_date));
373 do {
374 cout << "Sports: ";
375 cin << name;
376 if (validateSports(name, data.born_date)) {
377 errorInput();
378 }
379 } while (!validateSports(name, data.born_date));
380 do {
381 cout << "Games: ";
382 cin << name;
383 if (validateGames(name, data.born_date)) {
384 errorInput();
385 }
386 } while (!validateGames(name, data.born_date));
387 do {
388 cout << "Hobbies: ";
389 cin << name;
390 if (validateHobbies(name, data.born_date)) {
391 errorInput();
392 }
393 } while (!validateHobbies(name, data.born_date));
394 do {
395 cout << "Interests: ";
396 cin << name;
397 if (validateInterests(name, data.born_date)) {
398 errorInput();
399 }
400 } while (!validateInterests(name, data.born_date));
401 do {
402 cout << "Skills: ";
403 cin << name;
404 if (validateSkills(name, data.born_date)) {
405 errorInput();
406 }
407 } while (!validateSkills(name, data.born_date));
408 do {
409 cout << "Languages: ";
410 cin << name;
411 if (validateLanguages(name, data.born_date)) {
412 errorInput();
413 }
414 } while (!validateLanguages(name, data.born_date));
415 do {
416 cout << "Travel: ";
417 cin << name;
418 if (validateTravel(name, data.born_date)) {
419 errorInput();
420 }
421 } while (!validateTravel(name, data.born_date));
422 do {
423 cout << "Food: ";
424 cin << name;
425 if (validateFood(name, data.born_date)) {
426 errorInput();
427 }
428 } while (!validateFood(name, data.born_date));
429 do {
430 cout << "Drinks: ";
431 cin << name;
432 if (validateDrinks(name, data.born_date)) {
433 errorInput();
434 }
435 } while (!validateDrinks(name, data.born_date));
436 do {
437 cout << "Clothing: ";
438 cin << name;
439 if (validateClothing(name, data.born_date)) {
440 errorInput();
441 }
442 } while (!validateClothing(name, data.born_date));
443 do {
444 cout << "Accessories: ";
445 cin << name;
446 if (validateAccessories(name, data.born_date)) {
447 errorInput();
448 }
449 } while (!validateAccessories(name, data.born_date));
450 do {
451 cout << "Grooming: ";
452 cin << name;
453 if (validateGrooming(name, data.born_date)) {
454 errorInput();
455 }
456 } while (!validateGrooming(name, data.born_date));
457 do {
458 cout << "Beauty: ";
459 cin << name;
460 if (validateBeauty(name, data.born_date)) {
461 errorInput();
462 }
463 } while (!validateBeauty(name, data.born_date));
464 do {
465 cout << "Fitness: ";
466 cin << name;
467 if (validateFitness(name, data.born_date)) {
468 errorInput();
469 }
470 } while (!validateFitness(name, data.born_date));
471 do {
472 cout << "Sports: ";
473 cin << name;
474 if (validateSports(name, data.born_date)) {
475 errorInput();
476 }
477 } while (!validateSports(name, data.born_date));
478 do {
479 cout << "Games: ";
480
```

#### Pertemuan ke-4: POST TEST – Link List



Fungsi `input()` adalah sebuah fungsi dalam bahasa pemrograman C++ yang bertujuan untuk menambahkan data pada linked list yang merepresentasikan data-data pada sebuah Kartu Tanda Penduduk (KTP). Fungsi ini menambahkan data baru pada linked list dari belakang (`InsertAtTail`).

Pertama-tama, fungsi ini membuat sebuah simpul baru dengan menggunakan operator `new` dan menyimpan alamat simpul tersebut pada variabel `newNode`. Selanjutnya, fungsi ini meminta pengguna untuk memasukkan data-data pada KTP seperti provinsi, kota, NIK, nama, tempat lahir, tanggal lahir, bulan lahir, tahun lahir, jenis kelamin, golongan darah, alamat, nomor RT, nomor RW, kelurahan, kecamatan, agama, status pernikahan, dan pekerjaan. Setiap data yang dimasukkan akan divalidasi terlebih dahulu menggunakan fungsi-fungsi validasi yang telah didefinisikan sebelumnya.

Setelah semua data telah dimasukkan dan divalidasi, fungsi ini akan menambahkan simpul baru tersebut pada linked list dari belakang (`InsertAtTail`). Pertama-tama, simpul baru tersebut akan dihubungkan dengan simpul terakhir pada linked list dengan cara mengatur variabel `next` pada simpul terakhir agar menunjuk ke simpul baru tersebut. Selanjutnya, simpul baru tersebut akan dijadikan simpul terakhir pada linked list dengan cara mengatur variabel `head` agar menunjuk ke simpul baru tersebut. Terakhir, fungsi ini akan menambahkan nilai variabel `count` dengan 1 untuk menandakan bahwa jumlah simpul pada linked list telah bertambah.

Selain menambahkan simpul pada linked list, fungsi ini juga akan menuliskan data yang dimasukkan ke dalam file "data.txt" dengan format yang telah ditentukan.

```

100 void output() {
101 cout << "=====KARTU TANDA PENDUDUK===== << endl;
102 cout << "data: << endl;
103 cout << "===== << endl;
104 Node* current = head;
105 while (current != nullptr) {
106 int provincelength = current->data.province.length();
107 int citylength = current->data.city.length();
108 int maxlength = provincelength > citylength ? provincelength : citylength;
109 for (int j = 0; j < (maxlength - provincelength) / 2; j++) {
110 cout << " ";
111 }
112 cout << " " << current->data.province << endl;
113 for (int j = 0; j < (maxlength - citylength) / 2; j++) {
114 cout << " ";
115 }
116 cout << " " << current->data.city << endl;
117 cout << "NIK " << current->data.nik << endl;
118 cout << "Nama " << current->data.name << endl;
119 cout << "Tempat/Tgl Lahir " << current->data.born_place << ", " << 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;
120 cout << "Jenis Kelamin " << current->data.gender << " (laki-laki / perempuan)";
121 cout << setfill(' ') << setw(10) << current->data.bloodtype.length() << " Gol. Darah : " << current->data.bloodtype << endl;
122 cout << "Alamat " << current->data.address.address << endl;
123 cout << "RT/RW " << setfill('0') << setw(3) << current->data.address.neighborhoodAssociation << "/" << setfill('0') << setw(3) << current->data.address.communityAssociation << endl;
124 cout << "Kelurahan " << current->data.address.subdistrict << endl;
125 cout << "Kecamatan " << current->data.address.district << endl;
126 cout << "Agama " << current->data.religion << endl;
127 cout << "Status Perkawinan " << current->data.married;
128 cout << setfill(' ') << setw(20) << "Foto: jpg" << endl;
129 cout << "Pekerjaan " << current->data.job << endl;
130 cout << "Keanggotaan " << current->data.citizenship;
131 cout << setfill(' ') << setw(10) << current->data.created.data.length() << " " << setfill('0') << setw(2) << current->data.created.data << "-" << setfill('0') << setw(2) << current->data.created.month << "-" << setfill('0') << setw(2) << current->data.created.year << endl;
132 cout << "Morals Rating " << current->data.validity;
133 cout << setfill(' ') << setw(10) << "ID" << endl;
134 }
135 cout << "===== << endl;
136 current = current->next;
137 }

```

### Menampilkan data dengan prosedur `output()` pada class KTP

Fungsi `output()` adalah sebuah fungsi dalam bahasa pemrograman C++ yang bertujuan untuk menampilkan data-data pada linked list yang merepresentasikan data-data pada sebuah Kartu Tanda Penduduk (KTP) ke dalam output console dengan format yang telah ditentukan. Fungsi ini akan menampilkan data-data pada setiap simpul pada linked list secara berurutan.

Pertama-tama, fungsi ini akan menampilkan header yang berisi judul "KARTU TANDA PENDUDUK" ke dalam output console. Selanjutnya, fungsi ini akan menginisialisasi

variabel `current` dengan alamat simpul pertama pada linked list. Selama variabel `current` tidak bernilai `NULL`, fungsi ini akan menampilkan data-data pada simpul tersebut ke dalam output console dengan format yang telah ditentukan.

Setiap data pada simpul akan ditampilkan dengan format yang telah ditentukan. Beberapa data seperti provinsi, kota, jenis kelamin, dan golongan darah akan ditampilkan dengan format yang simetris dengan menggunakan fungsi `setfill()` dan `setw()`. Data tanggal lahir, nomor RT, nomor RW, dan tanggal pembuatan KTP akan ditampilkan dengan format yang telah ditentukan. Terakhir, fungsi ini akan menampilkan garis pemisah untuk memisahkan antara data pada simpul yang satu dengan simpul yang lain.

Fungsi ini tidak mengembalikan nilai apapun dan hanya menampilkan data-data pada linked list ke dalam output console dengan format yang telah ditentukan.





```

413 void delAtHead(){
414 if(head == nullptr){
415 cout << "Data kosong" << endl;
416 return;
417 }
418 Node* temp = head;
419 head = head->next;
420 delete temp;
421 cout << "Data berhasil dihapus" << endl;
422 count--;
423
424 ifstream file;
425 file.open("data.txt");
426 ofstream tempFile;
427 tempFile.open("temp.txt");
428 string line;
429 int lineCount = 0;
430 while (getline(file, line)) {
431 if (line.find("NIK") != string::npos) {
432 lineCount++;
433 }
434 if (lineCount != 1) {
435 tempFile << line << endl;
436 }
437 }
438 file.close();
439 tempFile.close();
440 remove("data.txt");
441 rename("temp.txt", "data.txt");
442 }
443 void delAtTail(){
444 if(head == nullptr){
445 cout << "linked list kosong" << endl;
446 return;
447 }
448 Node* current = head;
449 Node* prev = nullptr;
450 while(current->next != nullptr){
451 prev = current;
452 current = current->next;
453 }
454 if(prev == nullptr){
455 head = nullptr;
456 } else {
457 prev->next = nullptr;
458 }
459 delete current;
460 cout << "Data berhasil dihapus" << endl;
461 count--;
462
463 ifstream file;
464 file.open("data.txt");
465 ofstream tempFile;
466 tempFile.open("temp.txt");
467 string line;
468 int lineCount = 0;
469 while (getline(file, line)) {
470 if (line.find("NIK") != string::npos) {
471 lineCount++;
472 }
473 if (lineCount != count + 1) {
474 tempFile << line << endl;
475 }
476 }
477 file.close();
478 tempFile.close();
479 remove("data.txt");
480 rename("temp.txt", "data.txt");
481 }

```

#### Prosedur delAtHead() dan delAtTail() pada kelas KTP

Fungsi delAtHead() dan delAtTail() adalah fungsi dalam bahasa pemrograman C++ yang bertujuan untuk menghapus simpul pada linked list yang merepresentasikan data-data pada sebuah Kartu Tanda Penduduk (KTP). Fungsi delAtHead() akan menghapus simpul pertama pada linked list (delete di depan), sedangkan fungsi delAtTail() akan menghapus simpul terakhir pada linked list (delete di belakang).

Fungsi `delAtHead()` akan memeriksa apakah linked list kosong atau tidak. Jika linked list kosong, maka fungsi ini akan menampilkan pesan "Data kosong" ke dalam output console dan langsung keluar dari fungsi. Jika linked list tidak kosong, maka fungsi ini akan menghapus simpul pertama pada linked list dengan cara mengatur variabel head agar menunjuk ke simpul kedua pada linked list. Selanjutnya, fungsi ini akan menghapus simpul pertama tersebut dengan menggunakan operator delete. Terakhir, fungsi ini akan mengurangi nilai variabel count dengan 1 untuk menandakan bahwa jumlah simpul pada linked list telah berkurang.

Selain menghapus simpul pada linked list, fungsi `delAtHead()` juga akan menghapus data pada file "data.txt" yang sesuai dengan simpul yang dihapus. Fungsi ini akan membuka file "data.txt" dan membuat file sementara "temp.txt". Selanjutnya, fungsi ini akan membaca setiap baris pada file "data.txt" dan menuliskan baris tersebut ke dalam file sementara "temp.txt" kecuali baris yang sesuai dengan simpul yang dihapus. Setelah selesai, fungsi ini akan menutup kedua file dan menghapus file "data.txt" serta mengganti namanya dengan "temp.txt".

Fungsi `delAtTail()` akan memeriksa apakah linked list kosong atau tidak. Jika linked list kosong, maka fungsi ini akan menampilkan pesan "Linked list kosong" ke dalam output console dan langsung keluar dari fungsi. Jika linked list tidak kosong, maka fungsi ini akan mencari simpul terakhir pada linked list dengan cara mengiterasi dari simpul pertama hingga simpul terakhir. Setelah simpul terakhir ditemukan, fungsi ini akan menghapus simpul tersebut dengan cara mengatur variabel next pada simpul sebelum simpul terakhir agar menunjuk ke NULL. Terakhir, fungsi ini akan menghapus simpul terakhir tersebut dengan menggunakan operator delete. Fungsi ini juga akan mengurangi nilai variabel count dengan 1 untuk menandakan bahwa jumlah simpul pada linked list telah berkurang.

Selain menghapus simpul pada linked list, fungsi `delAtTail()` juga akan menghapus data pada file "data.txt" yang sesuai dengan simpul yang dihapus. Fungsi ini akan membuka file "data.txt" dan membuat file sementara "temp.txt". Selanjutnya, fungsi ini akan membaca setiap baris pada file "data.txt" dan menuliskan baris tersebut ke dalam file sementara "temp.txt" kecuali baris yang sesuai dengan simpul yang dihapus. Setelah selesai, fungsi ini akan menutup kedua file dan menghapus file "data.txt" serta mengganti namanya dengan "temp.txt".

```

483
484 int main() {
485 KTP ktp;
486 int choice;
487 while(true){
488 ktp.initialize();
489 cout << "Menu: " << endl;
490 cout << "1. Input Data" << endl;
491 cout << "2. Tampilkan Data" << endl;
492 cout << "3. Delete Data" << endl;
493 cout << "4. Keluar" << endl;
494 cout << "Pilihan: ";
495 cin >> choice;
496 system("cls");
497 cin.ignore();
498 switch (choice) {
499 case 1:
500 ktp.input();
501 cout << "Data berhasil diinput" << endl;
502 break;
503 case 2:
504 ktp.output();
505 break;
506 case 3:
507 while(true){
508 system("cls");
509 cout << "Menu: " << endl;
510 cout << "1. Delete data di depan" << endl;
511 cout << "2. Delete data di belakang" << endl;
512 cout << "3. Kembali" << endl;
513 cout << "Pilihan: ";
514 cin >> choice;
515 switch(choice){
516 case 1:
517 ktp.delAtHead();
518 break;
519 case 2:
520 ktp.delAtTail();
521 break;
522 case 3:
523 break;
524 default:
525 cout << "Pilihan tidak valid" << endl;
526 system("pause");
527 break;
528 }
529 break;
530 }
531 break;
532 case 4:
533 return EXIT_SUCCESS;
534 default:
535 cout << "Pilihan tidak valid" << endl;
536 break;
537 }
538 system("pause");
539 system("cls");
540 }
541 }
542 }

```

### Fungsi Utama

Fungsi main() adalah fungsi utama dalam bahasa pemrograman C++ yang akan dieksekusi pertama kali ketika program dijalankan. Fungsi ini bertanggung jawab untuk mengatur alur program dan memanggil fungsi-fungsi yang diperlukan.

Pada program ini, fungsi `main()` akan membuat sebuah objek dari kelas KTP dengan menggunakan sintaks `KTP ktp;`. Selanjutnya, fungsi ini akan menampilkan menu pilihan yang terdiri dari 4 pilihan yaitu input data, tampilkan data, delete data, dan keluar. Pilihan yang dimasukkan oleh pengguna akan disimpan pada variabel `choice`.

Setelah pengguna memilih salah satu pilihan, fungsi `main()` akan memanggil fungsi yang sesuai dengan pilihan pengguna. Jika pengguna memilih pilihan 1, maka fungsi `main()` akan memanggil fungsi `input()` dari objek `ktp` untuk memasukkan data baru pada linked list. Jika pengguna memilih pilihan 2, maka fungsi `main()` akan memanggil fungsi `output()` dari objek `ktp` untuk menampilkan data-data pada linked list ke dalam output console. Jika pengguna memilih pilihan 3, maka fungsi `main()` akan menampilkan sub-menu pilihan untuk delete data di depan atau di belakang. Pilihan yang dimasukkan oleh pengguna akan disimpan pada variabel `choice`. Jika pengguna memilih pilihan 1, maka fungsi `main()` akan memanggil fungsi `delAtHead()` dari objek `ktp` untuk menghapus simpul pertama pada linked list. Jika pengguna memilih pilihan 2, maka fungsi `main()` akan memanggil fungsi `delAtTail()` dari objek `ktp` untuk menghapus simpul terakhir pada linked list. Jika pengguna memilih pilihan 3, maka fungsi `main()` akan kembali ke menu utama. Jika pengguna memilih pilihan 4, maka fungsi `main()` akan keluar dari program.

Setelah memanggil fungsi yang sesuai dengan pilihan pengguna, fungsi `main()` akan menampilkan pesan "Data berhasil diinput" atau "Data berhasil dihapus" ke dalam output console jika operasi input atau delete berhasil dilakukan. Jika pengguna memasukkan pilihan yang tidak valid, maka fungsi `main()` akan menampilkan pesan "Pilihan tidak valid" ke dalam output console.

Setelah menampilkan pesan, fungsi `main()` akan menunggu pengguna menekan tombol "Enter" untuk melanjutkan program.

Untuk mengakses kodingan dan mencoba kodingan, dapat melihat source code berikut:

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