|  |
| --- |
| Aplikasi Pendataan Apotek Bandung |
| Laporan Tugas Besar Algoritma Struktur Data |
| Dosen Pengampu: Shinta Yulia Puspitasari, S.T  Kelompok:  Anggota Kelompok:   1. Dani Agung Prastiyo 1301154646 2. Hermawan Wiwit N. 1301154247 |

11/30/2016

**APOTEKPEDIA**

# Kelompok:

|  |  |  |
| --- | --- | --- |
| **No** | **NIM** | **Nama** |
| 1 | 1301154646 | Dani Agung Prastiyo |
| 2 | 1301154247 | Hermawan Wiwit N. |

# Penjelasan Program

Apotekpedia adalah aplikasi yang berfungsi untuk melakukan pendataan apotek-apotek yang berada di Kabupaten Bandung. Program ini melakukan pendataan dengan representasi multi linked list yang mana list tersebut mempunyai list induk dan list anak. List induk berisi daftar nama apotek yang mempunyai atribut ID Apotek, alamat apotek, jumlah obat yang dijual. . Sedangkan list anak berisi daftar menu obat yang dijual apotek tersebut, mempunyai atribut ID obat, nama obat, jenis obat, tanggal kadaluarsa, stok obat yang tersedia dan harga dari obat tersebut.

# Spesifikasi Program

Program ini dibagun dengan menggunakan spesifikasi sebagai dengan batasan berikut :

1. Bahasa pemograman yang digunakan adalah Bahasa pemograman C++, dan menggunakan tools Code Blocks.
2. Program ini berjalan di sistem operasi Windows dari Windows 7 sampai 10.
3. Program ini berjalan dalam desktop dengan tampilan *Command Line Interfaces* (CLI).
4. Program ini menerapkan Algoritma Multi Linked List.

# Fitur Program

Program ini memiliki fitur-fitur utama sebagai berikut:

1. Insert data apotek

Insert data apotek ini berarti menginputkan data sebuah apotek baru. Dalam insert ini user akan menginputkan ID apotek, nama apotek dan alamat apotek, setelah user menginputkan data apotek tersebut, lalu user memilih apakah akan melakukan insert first, insert after, atau insert last.

1. Insert obat apotek

Insert obat apotek ini berarti menginputkan data obat dari sebuah apotek. Saat insert, user akan memilih terlebih dahulu apotek mana yang akan diinputkan data obat baru. Setelah itu, user akan menginputkan data obat berupa ID obat, nama obat, jenis obat, tanggal kadaluarsa, stok obat yang tersedia dan harga dari obat tersebut. Terakhir, user akan memilih, apakah akan inset first, insert after, atau insert last.

1. Edit Info Apotek

Dalam edit info apotek ini, user akan mengedit data-data apotek yang telah diinputkan sebelumnya. Dengan memilih atau memasukkan id apotek untuk pertama kali memilih.

1. Edit info obat

Dalam edit info obat ini, user akan mengedit data-data obat yang berada di suatu apotek yang telah diinputkan sebelumnya. Dengan memilih atau memasukkan id apotek untuk pertama kali memilih. Lalu memilih atau memasukkan id obat yang akan diedit datanya.

1. Delete apotek

Menghapus data apotek dengan terlebih dahulu memilih apakah akan menerapkan delete first, delete last, atau delete after, selain ini delete juga bisa dilakukan berdasarkan dengan memasukkan id apotek yang akan dihapus.

1. Delete Obat Apotek

Delete obat apotek berarti menghapus data obat dari suatu apotek dengan terlebih dahulu memilih apakah akan menerapkan delete first, delete last, atau delete after, selain ini delete juga bisa dilakukan berdasarkan dengan memasukkan id apotek yang akan dihapus.

1. Print info Apotek

Print info Apotek ini berarti menampilkan list apotek yang tersedia beserta data-data apotek yang telah diinputkan sebelumnya.

1. Print info Apotek dan obatnya

Print info Apotek dan obatnya ini berarti menampilkan list apotek yang tersedia beserta data apotek tersebut lalu diikuti menampilkan data-data obat yang dimiliki apotek tersebut.

1. Search Apotek

Dalam pencarian apotek ini user dapat menginputkan ID apotek atau nama apotek yang dicari. Jika apotek ditemukan, maka program menampilkan data dari apotek tersebut.

1. Search Obat Apotek

Dalam pencarian obat dari suatu apotek ini user dapat menginputkan ID obat atau nama obat yang dicari. Jika obat ditemukan, maka program menampilkan data dari obat tersebut. Program juga menampilkan obat tersebut tersedia di apotek mana saja.

1. Sorting Apotek dengan ID

Sorting apotek dengan ID berarti program akan mengurutkan daftar apotek berdasarkan ID apotek yang telah di inputkan oleh user.

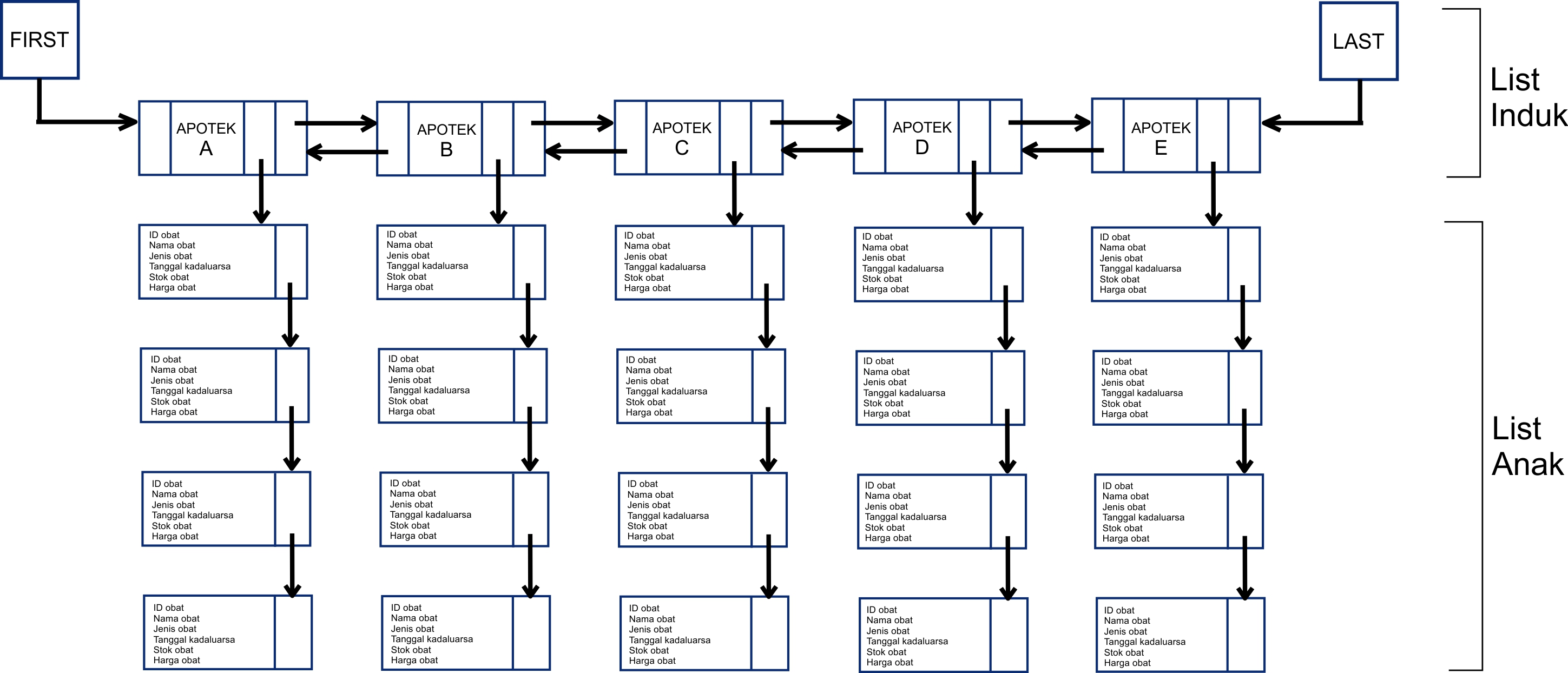
1. Sorting Obat Suatu APotek dengan ID

Sorting Obat dengan ID berarti program akan mengurutkan daftar obat berdasarkan ID obat yang telah di inputkan oleh user.

1. Hitung Total Apotek atau Obat

Dlam menu ini, program akan melakukan perhitungan terhadap daftar apotek yang telah diinputkan oleh user. Atau user juga bisa melihat jumlah obat yang dimiliki oleh apotek tertentu dengan memasukkan id apotek yang akan dilihat jumlah obatnya.

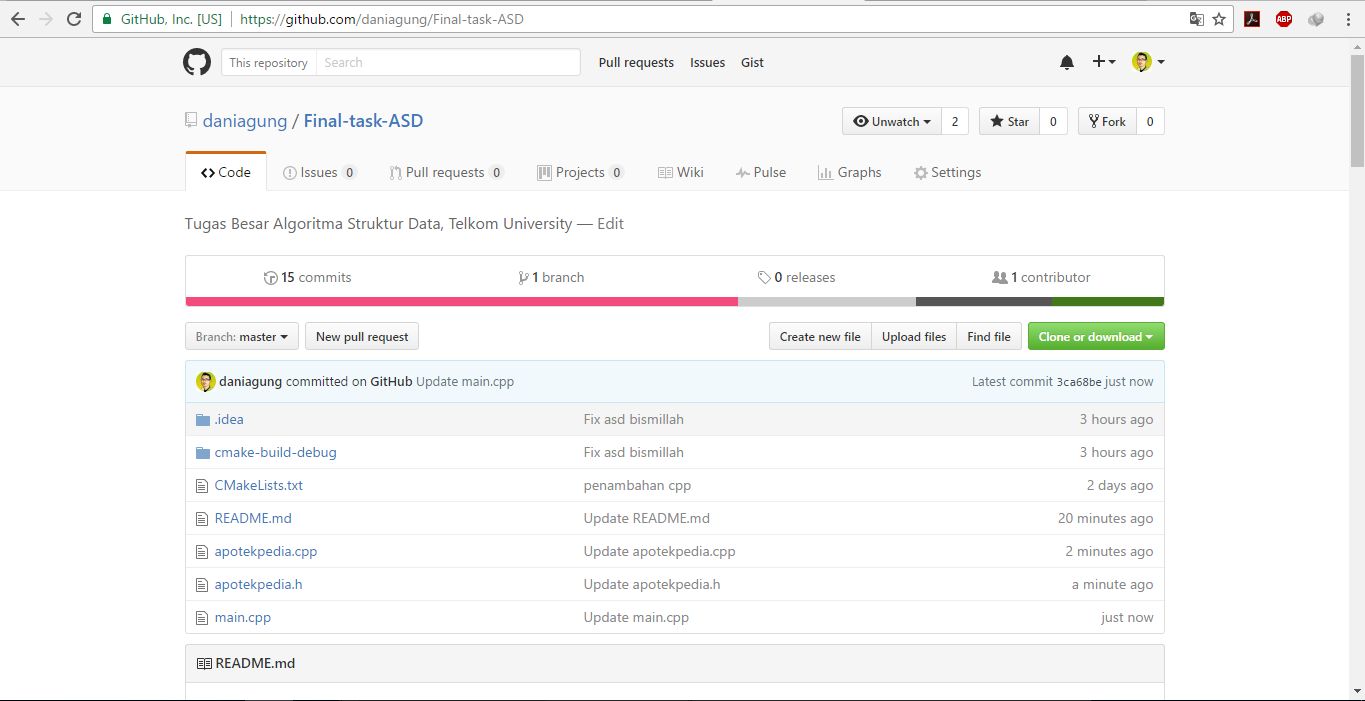
# Gambaran Dalam Program



# Dokumentasi Github

Dapat diakses di <https://github.com/daniagung/Final-task-ASD>

Link repository : https://github.com/daniagung/Final-task-ASD.git



# Cara Kerja Program

**Header : Apotekpediah.h**

**Define Pointer**

// Define Pointer  
#define info(P) P->info  
#define next(P) P->next  
#define prev(P) P->prev  
#define first(R) (R).first  
#define last(R) (R).last

**Deklarasi Pointer Address dari Parent Apotek**

**typedef int** \*addressAnggota;  
**typedef struct** elmAPK \*addressAPK;  
**typedef struct** elmObat \*addressObat;

**Deklarasai Info Bentukan Infotype Parent Apotek**

**struct** infotypeAPK  
{  
 string idAPK;  
 string namaAPK;  
 string alamat;  
 string tanggal;  
};

**Deklarasi Info Bentukan Infotype Child Obat**

**struct** infotypeObat {  
 string idObat;  
 string namaObat;  
 string kadaluarsa;  
 string stock;  
 string hargaObat;  
 string jenisObat;  
 addressAnggota anggota;  
};

**Tipe Bentukan Elemen List**

//Tipe bentukan Parent Apotek  
**struct** elmAPK  
{  
 infotypeAPK info;  
 addressObat daftarObat;  
 addressAPK next;  
 addressAPK prev;  
};  
  
//Tipe bentukan Child Obat  
**struct** elmObat {  
 infotypeObat info;  
 addressObat next;  
};

**Tipe Bentukan List**

**struct** apotek  
{  
 addressAPK first;  
 addressAPK last;  
};  
  
**struct** Obat {  
 addressObat first;  
};

**Fungsi dan Prosedur**

**void** createListAPK(apotek &R);  
**void** createAPK(infotypeAPK &APK);  
**void** createObat(infotypeObat &ObatAPK);  
addressAPK alokasiAPK(infotypeAPK APK);  
addressObat alokasiObat(infotypeObat ObatAPK);  
  
addressAPK findElmAPK(apotek R, string idAPK);  
addressObat findElmObat(addressAPK PAPK, string idObat);  
  
**void** printInfoAPK(apotek APK);  
**void** printInfoObat(apotek APK);  
**void** editInfoAPK(apotek &APK);  
**void** editHargaObat(apotek &APK);  
**void** searchAPK(apotek APK);  
**void** searchObat(apotek APK);  
  
**void** insertObatFirst(addressAPK PAPK, infotypeObat addObat);  
**void** insertObatLast(addressAPK PAPK, infotypeObat addObat);  
**void** insertObatAfter(addressAPK PAPK, infotypeObat addObat);  
**void** addObatAPK(apotek &APK);  
  
addressAPK deleteFirstAPK(apotek &APK);  
**void** deleteLastAPK(apotek &APK);  
**void** deleteAfterAPK(apotek &APK);  
**void** deleteAPKbyID(apotek &APK);  
**void** deleteAPKOption(apotek &APK);  
  
**void** deleteObatFirst(apotek &APK);  
**void** deleteObatAfter(apotek &APK);  
**void** deleteObatLast(apotek &APK);  
**void** deleteObatByID(apotek &APK);  
**void** deleteObatOption(apotek &APK);  
  
**void** insertFirstAPK(apotek &R, addressAPK P);  
**void** insertLastAPK(apotek &R, addressAPK P);  
**void** insertAfterAPK(apotek &R, addressAPK P);  
**void** addapotek(apotek &R);  
  
**void** countTotalAPK(apotek APK);  
**void** countTotalObat(apotek APK);  
**void** menucountObat(apotek APK);  
  
**int** countLengthObat(addressObat PAPK);  
**void** sortapotek(apotek &APK);  
**void** sortObat(apotek &APK);  
  
  
addressObat deleteObatFirstMergeSort(addressObat &M);  
**void** insertObatFirstMergeSort(addressObat &PAPK, addressObat M);  
**void** insertObatLastMergeSort(addressObat &PAPK, addressObat P);  
addressObat mergeInObat(addressObat left, addressObat right);  
addressObat mergeSortObat(addressObat PAPK);  
**void** sortObat(apotek &APK);

**Source File Apotekpedia.cpp**

**Library Yang Digunakan**

#include <iostream>  
#include <conio.h>  
#include <windows.h>  
#include <ctime>  
#include <cstdio>  
#include <sstream>

**Create List**

**void** createListAPK(apotek &R) {  
 first(R)= NULL;  
 last(R)= NULL;  
}

Membuat List yang di inputkan di parameter menjadi kosong dengan memberi nilai NULL pada first dan last pada list tersebut.

**Create List Parent Apotek**

**void** createAPK(infotypeAPK &APK) {  
  
 cout<<"== Input Apotek Baru =="<<endl;  
 cout<<"Input ID APK : "; getline(cin, APK.idAPK);  
 cout<<"Nama Apotek : "; getline(cin, APK.namaAPK);  
 cout<<"Alamat Apotek : "; getline(cin, APK.alamat);  
 time\_t now = time(0);  
 string dt = ctime(&now);  
 cout<<"Tanggal Input : "<<dt;  
 APK.tanggal=dt;

Meminta Inputan User sesuai dengan atribut dari Apotek (Elemen Parent) untuk disimpan pada variabel yang terdapat pada parameter.

**Create List Child Obat**

**void** createObat(infotypeObat &ObatAPK) {  
 cout<<"== Input Obat Baru =="<<endl;  
 cout<<"Input ID Obat : "; getline(cin, ObatAPK.idObat);  
 cout<<"Nama Obat : "; getline(cin, ObatAPK.namaObat);  
 cout<<"Jenis Obat : "; getline(cin, ObatAPK.jenisObat);  
 cout<<"Stock Obat : "; getline (cin, ObatAPK.stock);  
 cout<<"Kadaluarsa : "; getline(cin, ObatAPK.kadaluarsa);  
 cout<<"Harga (NUM) : "; cin>>ObatAPK.hargaObat; cin.ignore();  
  
 cout<<endl;  
}

Meminta Inputan User sesuai dengan atribut dari Obat (Elemen Child) untuk disimpan pada variabel yang terdapat pada parameter.

**Alokasi Elemen Parent (Apotek)**

addressAPK alokasiAPK(infotypeAPK APK) {  
 addressAPK P;  
 P = **new** elmAPK;  
 info(P) = APK;  
 next(P) = NULL; //belum tersambung  
 prev(P) = NULL; //belum tersambung  
 P->daftarObat = NULL; //belum tersambung  
 **return** P;  
}

Mengalokasikan Elemen Parent (Apotek) yang belum tersambung dengan elemen manapun sesuai dengan infotype yang ada dalam parameter dan

**Alokasi Elemen Child (Obat)**

addressObat alokasiObat(infotypeObat ObatAPK) {  
 addressObat Q = **new** elmObat;  
 Q->info = ObatAPK;  
 Q->next = NULL;  
 **return** Q;  
}

Mengalokasikan Elemen Parent (Apotek) yang belum tersambung dengan elemen manapun sesuai dengan infotype yang ada dalam parameter dan

**Searching Element di List Apotek (Metode Sequential Search)**

addressAPK findElmAPK(apotek R, string idAPK) {  
 addressAPK P;  
 **if**(first(R) == NULL)  
 {  
 **return** NULL;  
 }  
 **else** {  
 **bool** ketemu = **false**;   
 P = first(R);  
 **while** ((info(P).idAPK != idAPK) && (ketemu == **false**))  
 {  
 **if** (next(P) == NULL) {  
 ketemu = **true**;  
 }  
 **else** {  
 P = next(P);  
 }  
 }  
 **if** (ketemu)  
 **return** NULL;  
 **else  
 return** P;  
 }  
}

Mencari elemen pada list parent (apotek) yang Id-nya sesuai dengan inputan yang ada pada parameter.

**Prosedur Print Info Parent Apotek**

**void** printInfoAPK(apotek APK){  
 addressAPK P;  
 P = APK.first;  
 **if**(APK.first == NULL)  
 cout<<"Tidak ada Apotek yang terdaftar"<<endl;  
 **else**{  
 cout<<"========== List Apotek yang Tersedia =========="<<endl;  
 **int** i = 1;  
 **while**(P != NULL){  
  
 cout<<"Data Apotek Ke-"<<i<<endl; i++;  
 cout<<"ID Apotek : "<<P->info.idAPK<<endl;  
 cout<<"Nama Apotek : "<<P->info.namaAPK<<endl;  
 cout<<"Alamat Apotek : "<<P->info.alamat<<endl;  
 cout<<"TGL Pendataan Apotek : "<<P->info.tanggal<<endl;  
 cout<<endl;  
 P = P->next;  
 }  
 }  
}

Menampilkan seluruh info pada seluruh element pada list parent (apotek). Jika list parent Apotek tidak tersedia maka program akan menampilkan list apotek kosong, jika list parent apotek ada data listnya maka program akan menampilkan seluruh list yang tersedia.

**Prosedur Print Info List Child (Obat)**

**void** printInfoObat(apotek APK){  
 addressAPK P;  
 addressObat Q;  
 P = APK.first;  
 **if**(APK.first == NULL)  
 cout<<"Tidak Ada Apotek Terdaftar"<<endl;  
 **else** {  
 cout<<"==== Data Apotek beserta Obatnya ===="<<endl;  
 cout<<endl;  
 **int** i = 1;  
 **while**(P != NULL){  
 cout<<"--- Data Apotek ke-"<<i<< " ---"<<endl; i++;  
 cout<<"ID Apotek : "<<P->info.idAPK<<endl;  
 cout<<"Nama Apotek : "<<P->info.namaAPK<<endl;  
 Q = P->daftarObat;  
 **if**(Q==NULL){  
 cout<<"\n====== Obat Kosong ======"<<endl;  
 }**else**{  
 cout<<"\n====== List Obat ======"<<endl;  
 }  
 **while**(Q != NULL){  
 cout<<"ID Obat : "<<Q->info.idObat<<endl;  
 cout<<"Nama Obat : "<<Q->info.namaObat<<endl;  
 cout<<"Jenis Obat : "<<Q->info.jenisObat<<endl;  
 cout<<"Harga Obat : "<<Q->info.hargaObat<<endl;  
 cout<<"Stock Obat : "<<Q->info.stock<<endl;  
 cout<<"Kadaluarsa : "<<Q->info.kadaluarsa<<endl;  
 cout<<endl;  
 Q = Q->next;  
 }  
 cout<<endl;  
 P = P->next;  
 }  
 }  
}

Menampilkan seluruh info pada seluruh element pada list child (obat). Jika list parent Apotek tidak tersedia maka program akan menampilkan list apotek kosong, jika list parent apotek ada data listnya maka program akan menampilkan seluruh list yang tersedia.

**Prosedur Edit Parent (Apotek)**

**void** editInfoAPK(apotek &APK){  
 **if**(APK.first != NULL){ //Cek Apotek Apakah Kosong atau Tidak  
 printInfoAPK(APK);  
 string strSearchAPK;  
 cout << "Masukkan ID Apotek yang mau di Edit : ";cin>>strSearchAPK; cin.ignore();  
 addressAPK PAPK = findElmAPK(APK, strSearchAPK); //Mencari Elemen Apotek  
 **if**(PAPK == NULL){  
 cout<<"ID apotek Tidak Ditemukan"<<endl;  
 cout<<endl;  
 } **else** {  
 cout<<"ID APK Ditemukan"<<endl;  
 cout<<"Edit Nama APK : "; getline(cin, PAPK->info.namaAPK);  
 cout<<"Edit Alamat APK : "; getline(cin, PAPK->info.alamat);  
 cout<<"Edit Successfull"<<endl;  
 cout<<endl;  
 }  
 } **else** {  
 cout<<"Tidak ada Apotek yang bisa di edit (Kosong) "<<endl;  
 }  
}

Meminta inputan user untuk meninputkan ID-Apotek dan mencari elemen di list dengan memanggil function Findelement untuk meng edit info ID dari element parent (apotek) yang di cari.

**Prosedur Edit Child (Obat)**

**void** editObat(apotek &APK) {  
 **if**(APK.first == NULL)  
 cout<<"Tidak ada Apotek yang Terdata (Kosong)"<<endl;  
 **else** {  
 printInfoAPK(APK);  
 string strSearchID;  
 string strSearchIDObat;  
 cout << "Masukkan ID Apotek Yang Akan Diubah Harga Obatnya : ";getline(cin, strSearchID);  
 addressAPK PAPK = findElmAPK(APK, strSearchID);  
 **if** (PAPK == NULL) {  
 cout << "Tidak Ditemukan Apotek";  
 }  
 **else** {  
 addressObat M = PAPK->daftarObat;  
 cout << "List Obat APK - "<<PAPK->info.namaAPK<<endl;  
 **while** (M != NULL){ // Looping Untuk Menampilkan Obat Apotek  
 cout<<"ID Obat : "<<M->info.idObat<<endl;  
 cout<<"Nama Obat : "<<M->info.namaObat<<endl;  
 cout <<"Jenis Obat : "<<M->info.jenisObat<<endl;  
 cout <<"Harga Obat : "<<M->info.hargaObat<<endl;  
 cout <<"Stock Obat : "<<M->info.stock<<endl;  
 cout<<endl;  
 M = M->next;  
 }  
 cout << endl;  
 cout << "Masukkan ID Obat Yang Harganya Akan Diubah : ";getline(cin, strSearchIDObat);  
 M = PAPK->daftarObat;  
 **while** (M != NULL) {  
 **if** (M->info.idObat == strSearchIDObat) {  
 **break**;  
 }  
 **else** {  
 M = M->next;  
 }  
 }  
 **if** (M == NULL) {  
 cout << "Tidak Ada Obat Yang Sesuai"<<endl;  
 }  
 **else** {  
 cout << "Obat Ditemukan"<<endl;  
 cout << "Nama Obat Baru : "<<M->info.namaObat<<endl;  
 cout << "Jenis Obat Baru : ";cin>>info.jenisObat<<endl;  
 cout << "Harga Obat Baru : ";cin>>M->info.hargaObat<<endl;  
 cout << "Stock Obat Baru : ";cin>>M->info.stock<<endl;  
 cout << "Masukkan Harga Baru : ";cin>>M->info.hargaObat;  
 cout << "Data Berhasil Di Ubah"<<endl;  
 }  
 }  
 }  
}

Meminta inputan user untuk meninputkan ID-Apotek, setelah itu menampilkan semua daftar obat yang ada dalam apotek tersebut, kemudia program meminta inputan ID-Obat apa yang akan di edit

**Prosedur Searching List Parent Apotek (Metode Sequential Search)**

**void** searchAPK(apotek APK){  
 **if** (APK.first == NULL)  
 cout <<"Tidak Ada Data Apotek"<<endl;  
 **else**{  
 string strSearch;  
 cout <<"Masukkan ID / Nama APK yang akan dicari : "; getline(cin, strSearch);  
 addressAPK P = APK.first;  
 **while** (P != NULL){  
 **if**((P->info.idAPK == strSearch) || (P->info.namaAPK == strSearch))  
 **break**;  
 **else** P = P->next;  
 }  
 **if** (P==NULL){  
 cout <<"Tidak ada Data yang Sesuai"<<endl;  
 cout <<endl;  
 }**else**{  
 cout <<"Data Ditemukan"<<endl;  
 cout <<"ID Apotek : "<<P->info.idAPK<<endl;  
 cout <<"Nama Apotek : "<<P->info.namaAPK<<endl;  
 cout <<"Alamat Apotek : "<<P->info.alamat<<endl;  
 cout<<"TGL Pendataan Apotek : "<<P->info.tanggal<<endl;  
 }  
 }  
}

Meminta user untuk menginputkan ID dari element parent (Apotek) yang dicari dan meng-outputkan info dari element tersebut jika ketemu.

**Prosedur Searching List Child Obat (Metode Sequential Search)**

**void** searchObat(apotek APK) {  
 **if** (APK.first == NULL)  
 cout << "Tidak ada Data Apotek" << endl;  
 **else** {  
 string strSearch;  
 cout << "Masukkan ID / Nama Obat yang akan dicari : ";  
 getline(cin, strSearch);  
 addressAPK P = APK.first;  
 addressObat Q;  
 **bool** checkFound = **false**;  
 **while** (P != NULL) {  
 Q = P->daftarObat;  
 **while** (Q != NULL) {  
 **if** ((Q->info.idObat == strSearch) || (Q->info.namaObat == strSearch)) {  
 checkFound = **true**;  
 **break**;  
 }  
 Q = Q->next;  
 }  
 **if** (checkFound)  
 **break**;  
 P = P->next;  
 }  
 **if** (Q == NULL) {  
 cout << "Tidak ada Data yang Sesuai" << endl;  
 cout << endl;  
 } **else** {  
 cout << "Data Ditemukan" << endl;  
 cout << "ID Obat : " << Q->info.idObat << endl;  
 cout << "Nama Obat : " << Q->info.namaObat << endl;  
 cout << "Jenis Obat : " << Q->info.jenisObat << endl;  
 cout << "Harga Obat : " << Q->info.hargaObat << endl;  
 cout << "Stock Obat : " << Q->info.stock << endl;  
 cout << "Kadaluarsa : " << Q->info.kadaluarsa << endl;  
 cout << "Terdapat di Apotek : " << P->info.namaAPK << endl;  
 cout << endl;  
 }  
 }  
}

Meminta user untuk menginputkan id obat, kemudian program akan mencari id obat pada elemen yang sesuai dengan inputan yang dicari. Kemudian jika ketemu program akan mengoutputkannya dan jika tidak program akan mengoutputkan list kosong.

**Prosedur Insert First Child (Obat)**

//----------------- Insert Obat ----------------//  
**void** insertObatFirst(addressAPK PAPK, infotypeObat addObat){  
 **if** (PAPK == NULL){  
 cout <<"Tidak Ditemukan Apotek";  
 }  
 **else**{  
 addressObat M = alokasiObat(addObat);  
 **if** (PAPK->daftarObat == NULL){ //PAPK itu Elemet parennya kalau daftar obat list childnya  
 PAPK->daftarObat = M;  
 }  
 **else** {  
 M->next = PAPK->daftarObat;  
 PAPK->daftarObat = M;  
 }  
 }  
}

Membuat elemen baru berdasarkan infotype yang ada dalam parameter kemudian meninsertkannya pada element parent yang ada di parameter pada urutan pertama.

**Prosedur Insert Last Child (Obat)**

**void** insertObatLast(addressAPK PAPK, infotypeObat addObat){  
 **if** (PAPK == NULL){  
 cout <<"Tidak Ditemukan Apotek";  
 }  
 **else** {  
 addressObat P;  
 P = PAPK->daftarObat;  
 addressObat M = alokasiObat(addObat);  
 **if** (P == NULL){  
 insertObatFirst(PAPK, addObat);  
  
 } **else** {  
 **while** (P->next != NULL){  
 P = P->next;  
 }  
 P->next = M;  
 }  
 }  
}

Membuat elemen baru berdasarkan infotype yang ada dalam parameter kemudian meninsertkannya pada element parent yang ada di parameter pada urutan terakhir.

Prosedur Insert After Child (Apotek) **void** insertObatAfter(addressAPK PAPK, infotypeObat addObat){  
 string searchIDObat;  
 addressObat M = PAPK->daftarObat;  
 **if** (M == NULL) {  
 cout << "karena tidak ada Obat yang Tersedia, maka langsung dimasukkan ke Awal List" <<endl;  
 insertObatFirst(PAPK, addObat);  
 }  
 **else** {  
 cout << "List Obat Apotek - "<<PAPK->info.namaAPK<<endl;  
 **while** (M != NULL){  
 //looping untuk menampilkan obat apotek  
 cout << "Data Ditemukan" << endl;  
 cout << "ID Obat : " << M->info.idObat << endl;  
 cout << "Nama Obat : " << M->info.namaObat << endl;  
 cout << "Jenis Obat : " << M->info.jenisObat << endl;  
 cout << "Harga Obat : " << M->info.hargaObat << endl;  
 cout << "Stock Obat : " << M->info.stock << endl;  
 cout << "Kadaluarsa : " << M->info.kadaluarsa << endl;  
 cout << endl;  
 M = M->next;  
 }  
 cout <<endl;  
 cout << "Masukkan ID Obat yang Ingin di Insert Sesudahnya: ";getline(cin, searchIDObat);  
 M = findElmObat(PAPK,searchIDObat);  
 **if** (M == NULL){  
 cout<<"elemen ysng di insert sesudahnya tidak tersedia"<<endl;  
 }  
 **else** {  
 addressObat addAPKSisip;  
 addAPKSisip = alokasiObat(addObat);  
 **if** (M->next == NULL){  
 //Jika Elemen di Ujung  
 insertObatLast(PAPK, addObat);  
 } **else** {  
 addAPKSisip->next = M->next;  
 M->next = addAPKSisip;  
 }  
 }  
 }  
}

Program mengoutputkan daftar obat yang ada di element parent yang ada di parameter kemudian meminta user untuk menginputkan Id dari elemen obat yang ingin di insertkan sesudahnya

**Prosedur Add Obat**

**void** addObatAPK(apotek &APK){  
 **if** (APK.first == NULL){  
 cout<<"Tidak Ada Apotek yang Terdaftar"<<endl;  
 } **else** {  
 string seacrhAPK;  
 printInfoAPK(APK);  
 cout<<"Masukkan ID Apotek yang akan ditambah Obat-nya: "; cin>>seacrhAPK; cin.ignore();  
 addressAPK PAPK = findElmAPK(APK, seacrhAPK);  
 **if** (PAPK == NULL){  
 cout<<"Tidak Ditemukan Apotek";  
 } **else** {  
 infotypeObat addObat;  
 createObat(addObat);  
 **char** menu;  
 cout<<endl;  
 cout<<"Pilih Jenis Insert Yang Anda Suka: "<<endl  
 <<"1. Insert First"<<endl  
 <<"2. Insert Last"<<endl  
 <<"3. Insert After"<<endl;  
 ObatLabel :  
 cout<<"Pilihan: ";cin>>menu; cin.ignore();  
 **switch** (menu){  
 **case** '1' : insertObatFirst(PAPK, addObat); **break**;  
 **case** '2' : insertObatLast(PAPK,addObat); **break**;  
 **case** '3' : insertObatAfter(PAPK,addObat); **break**;  
 **default** : cin.ignore(); **goto** ObatLabel;  
 }  
 cout<<"Data Berhasil di Insert"<<endl;  
 }  
 }  
}

Meminta inputan id dari parent (apotek) yang ingin ditambahkan obatnya kemudian meminta user untuk memilih jenis insert mana yg ingin dilakukan jika element parent (apotek) ada.

**Prosedure Delete First Apotek**

addressAPK deleteFirstAPK(apotek &APK){  
 addressAPK P;  
 **if** (first(APK) == NULL){  
 **return** NULL;  
 }  
 **else** {  
 P = first(APK);  
 **if** (next(first(APK))==NULL){  
 first(APK) = NULL;  
 last(APK) = NULL;  
 **return** P;  
 }  
 **else** {  
 prev(next(P)) = NULL;  
 first(APK) = next(P);  
 next(P) = NULL;  
 **return** P;  
 }  
 }  
}

Menghapus elemen pertama pada list parent (apotek). Jika list kosong tidak menghapus apa-apa dan jika list tersedia maka dia akan menghapus element parent yang pertama.

**Prosedur Delete Last Apotek**

**void** deleteLastAPK(apotek &APK){  
 addressAPK P;  
 **if** (first(APK) == NULL){  
 cout<<"Data Apotek tidak Tersedia"<<endl;  
 } **else** {  
 P = last(APK);  
 **if** (prev(P) == NULL){  
 first(APK) = NULL;  
 last(APK) = NULL;  
 **delete** P;  
 } **else** {  
 next(prev(P)) = NULL;  
 last(APK) = prev(P);  
 prev(P) = NULL;  
 }  
 }  
}

Menghapus elemen terakhir pada list parent (apotek). Jika list kosong tidak menghapus apa-apa dan jika list tersedia maka dia akan menghapus element parent yang terakhir.

**Prosedure Delete After Parent Apotek**

**void** deleteAfterAPK(apotek &APK){  
 addressAPK P;  
 **if** (first(APK) = NULL){  
 cout<<"Data Apotek Tidak ditemukan"<<endl;  
 } **else** {  
 string hapus;  
 cout<<endl;  
 printInfoAPK(APK);  
 cout<<"Masukkan ID Apotek Sebelum Apotek di depan Yang akan dihapus : ";  
 getline(cin, hapus);  
 addressAPK R = findElmAPK(APK,hapus);  
 **if** (R ->next == NULL){  
 cout<<"ini adalah Elemen Terakhir"<<endl;  
 } **else** {  
 addressAPK P = R->next;  
 **if** (P->next == NULL){ //elemen yang dihapus berada diujung  
 R->next = NULL;  
 P->prev = NULL;  
 **delete** P;  
 } **else** {  
 //elemen yang dihapus di tenggah-tenggah  
 R->next = P->next;  
 P->prev->prev = R;  
 P->next = NULL;  
 P->prev = NULL;  
 **delete** P;  
 }  
 cout<<"Delete Data Apotek Berhasih Yeah"<<endl;  
 }  
 }  
}

Program akan menghapus elemen dari elemen yang di inputkan id-nya jika elemen yang di inputkan idnya ada dan bukan elemen terakhir

**Prosedure Delete Option Parent Apotek**

**void** deleteAPKbyID(apotek &APK){  
 string searchAPK;  
 printInfoAPK(APK);  
 cout<<"Masukkan ID Apotek Sebelum Apotek di depan Yang akan dihapus : "; getline(cin, searchAPK);  
 addressAPK PAPK = findElmAPK(APK, searchAPK);  
 **if** (PAPK == NULL){  
 //ksus elemen tidak ditemukan  
 cout<<"ID Apotek tidak ditemukan"<<endl;  
 }  
 **else** {  
 **if** (APK.first->next == NULL){  
 APK.first = NULL;  
 APK.last = NULL;  
 **delete** PAPK;  
 }  
 **else if** ((PAPK == APK.last) && (PAPK->next != NULL)){  
 //Kasus elemen Yang di delete di elemen yang pertama  
 deleteFirstAPK(APK);  
 }  
 **else if** ((PAPK == APK.last) && (PAPK->prev != NULL)){  
 //Kasus Elemen yang di delete elemen terakhir  
 deleteLastAPK(APK);  
 }  
 **else if** ((PAPK != APK.first) && (PAPK != APK.last) && (PAPK->next != NULL)){  
 //kasus element di tenggah  
 addressAPK P =PAPK->prev;  
 PAPK->next = PAPK->next;  
 PAPK->next->prev = P;  
 PAPK->next = NULL;  
 PAPK->prev = NULL;  
 **delete** PAPK;  
 }  
 **else** cout<<"Tidak Dapat Di Delete"<<endl;  
  
 }  
}

Menghapus elemen parent sesuai dengan ID yang di inputkan, jika elemen berada pada urutan pertama di list memanggil delete first, jika elemen berada di urutan terakhir program memanggil delete last dan jika tidak keduanya maka program menghapus elemen dengan menggunakan konsep delete after.

**Prosedur Menu Delete Parent Apotek**

**void** deleteAPKOption(apotek &APK){  
 **if**(APK.first == NULL)  
 cout<<"Tidak ada Data Apotek (Kosong)"<<endl;  
 **else** {  
 **char** menu;  
 ObatLabel :  
 cout << "Pilih Jenis Metode Delete"<<endl  
 << "1. Pilih Delete First"<<endl  
 << "2. Pilih Delete Last"<<endl  
 << "3. Pilih Delete After"<<endl  
 << "4. Pilih Delete By ID"<<endl  
 << "5. Exit"<<endl;  
  
 cout <<"Masukkan Pilihan : ";  
 cin>>menu; cin.ignore();  
 **switch** (menu){  
 **case** '1' : deleteFirstAPK(APK); cout<<"Delete Data APK Berhasil"<<endl; **break**;  
 **case** '2' : deleteLastAPK(APK); cout<<"Delete Data APK Berhasil"<<endl; **break**;  
 **case** '3' : deleteAfterAPK(APK); cout<<"Delete Data APK Berhasil"<<endl; **break**;  
 **case** '4' : deleteAPKbyID(APK); cout<<"Delete Data APK Berhasil"<<endl; **break**;  
 **default** : cin.ignore(); **goto** ObatLabel;  
 }  
 //obatAkhir;  
 }  
}

Program mengoutputkan tipe-tipe delete yang bisa dilakukan dan program meminta user menginputkan delte yang mana yang di inputkan.

**Prosedur Delete Child First**

**void** deleteObatFirst(apotek &APK){  
 cout<<endl;  
 printInfoAPK(APK);  
 cout<<endl;  
 string SearchByID;  
 cout<<"Masukkan ID Apotek yang Akan Dihapus Obatnya: "; getline(cin,SearchByID);  
 addressAPK PAPK = findElmAPK(APK, SearchByID);  
 **if** (PAPK == NULL){  
 cout<<"Tidak Ditemukan Apotek";  
 }  
 **else** {  
 addressObat M = PAPK->daftarObat;  
 **if** (M == NULL){  
 cout<<"Tidak Terdapat Obat"<<endl;  
 }  
 **else** {  
 **if** (M->next == NULL){  
 PAPK->daftarObat = NULL;  
 M = NULL;  
 **delete** M;  
 }  
 **else** {  
 PAPK->daftarObat = M->next;  
 M->next = NULL;  
 **delete** M;  
 }  
 cout<<"Obat Berhasil dihapus"<<endl;  
 }  
 }  
}

Program meminta user untuk menginputkan id element parent yang ingin di hapus obatnya yang berada pada urutan pertama.

**Prosedure Delete Child Last**

**void** deleteObatLast(apotek &APK){  
 cout<<endl;  
 printInfoAPK(APK);  
 cout<<endl;  
 string SearchID;  
 cout << "Masukkan Id Apotek yang akan dihapus Obatnya: "; getline(cin, SearchID);  
 addressAPK PAPK = findElmAPK(APK, SearchID);  
 **if** (PAPK == NULL) {  
 cout <<"Tidak Ditemukan Apotek"<<endl;  
 }  
 **else**{  
 addressObat M = PAPK->daftarObat;  
 **if** (M == NULL){  
 cout << "Obat Kosong Pada Apotek Tersebut"<<endl;  
 }  
 **else** {  
 **while** (M->next != NULL){  
 M = M->next;  
 }  
 addressObat N = PAPK->daftarObat;  
 **while** (N->next->info.idObat != M->info.idObat){  
 N = N->next;  
 }  
 N->next = NULL;  
 **delete** M;  
 cout <<"Obat Berhasil Dihapus"<<endl;  
 }  
 }  
}

Program meminta user untuk menginputkan id element parent (apotek) yang ingin di hapus obatnya yang berada pada urutan pertama.

Prosedure Delete After Child (Obat)

**void** deleteObatAfter(apotek &APK){  
 cout<<endl;  
 printInfoAPK(APK);  
 cout<<endl;  
 string searchByIDAPK;  
 string SearchIDObat;  
 cout << "Masukkan ID Apotek yang akan di hapus Obatnya: ";getline(cin, searchByIDAPK);  
 addressAPK PAPK = findElmAPK(APK, searchByIDAPK);  
 **if** (PAPK == NULL){  
 cout<<"tidak Ditemukan Apotek"<<endl;  
 }  
 **else** {  
 addressObat M = PAPK->daftarObat;  
 **if** (M == NULL){  
 cout<<"Tidak Terdapat Obat"<<endl;  
 }  
 **else** {  
 cout << "List Obat APK - "<<PAPK->info.namaAPK<<endl;  
 **while** (M != NULL) {  
 //Loping untuk menapilkan Obat Apotek  
 cout << "Data Ditemukan" << endl;  
 cout << "ID Obat : " << M->info.idObat << endl;  
 cout << "Nama Obat : " << M->info.namaObat << endl;  
 cout << "Jenis Obat : " << M->info.jenisObat << endl;  
 cout << "Harga Obat : " << M->info.hargaObat << endl;  
 cout << "Stock Obat : " << M->info.stock << endl;  
 cout << "Kadaluarsa : " << M->info.kadaluarsa << endl;  
 cout << endl;  
 M = M->next;  
 }  
 cout <<endl;  
 cout<< "Masukkan ID Obat Sebelum ID yang akan Dihapus : ";  
 cin>> SearchIDObat; cin.ignore();  
 M = findElmObat(PAPK, SearchIDObat);  
 **if** ((M != PAPK->daftarObat) && (M->next == NULL)){  
 //elemen yang di ujung ya  
 cout<<"Tidak ada Obat lagi Setelah Obat ini"<<endl;  
 }  
 **else if** ((M == PAPK->daftarObat) && (M->next == NULL)) {  
 // Jika Obat Tunggal  
 cout<<"Obat Ini adalah Satu-satunya Obat Yang ada di Apotek"<<endl;  
 }  
 **else**{  
 addressObat N = M->next;  
 **if** (N->next == NULL ){  
 M->next = NULL;  
 **delete** N;  
 }  
 **else** {  
 M->next = N->next;  
 N->next = NULL;  
 **delete** N;  
 }  
 cout << "Obat Berhasil Dihapus"<<endl;  
 }  
 }  
 }  
}

Program meminta user untuk menginputkan Id elemen parent jika elemen terdapat dalam list maka program akan mengoutputkan obat-obat pada apotek yang di cari. Kemudian program meminta user menginputkan id obat yang ingin di hapus elemen sesudahnya.

**Prosedure Delete Child By ID**

**void** deleteObatByID(apotek &APK){  
 string SearchIDAPK;  
 string SearchIDObat;  
 printInfoAPK(APK);  
 cout <<"Masukkan ID Apotek yang akan dihapus Obatnya: "; getline(cin, SearchIDAPK);  
 addressAPK PAPK = findElmAPK(APK, SearchIDAPK); //mengambil keberadaan APK dimana  
 **if** (PAPK == NULL){  
 cout << "Tidak Ditemukan Apotek";  
 }  
 **else** {  
 addressObat M = PAPK->daftarObat;  
 **if** (M == NULL){  
 cout << "Tidak Terdapat Obat Dalam Apotek Tersebut"<<endl;  
 }  
 **else** {  
 cout << "List Obat APK - "<<PAPK->info.namaAPK<<endl;  
 **while** (M != NULL){ //Looping untuk menamoikan Obat Apotek  
 cout << "Data Ditemukan" << endl;  
 cout << "ID Obat : " << M->info.idObat << endl;  
 cout << "Nama Obat : " << M->info.namaObat << endl;  
 cout << "Jenis Obat : " << M->info.jenisObat << endl;  
 cout << "Harga Obat : " << M->info.hargaObat << endl;  
 cout << "Stock Obat : " << M->info.stock << endl;  
 cout << "Kadaluarsa : " << M->info.kadaluarsa << endl;  
 cout << endl;  
 M = M->next;  
 }  
 cout<<endl;  
 cout<< "Masukkan ID Obat Yang Ingin Dihapus : "; getline(cin, SearchIDObat);  
 //mencari ID Obat  
 M = PAPK->daftarObat;  
 **while** (M != NULL){  
 //looping untuk posisi elemnt yang dihapus  
 **if** (M->info.idObat == SearchIDObat){  
 **break**;  
 }  
 **else**{  
 M = M->next;  
 }  
 }  
 **if** (M == NULL){  
 cout <<"Tidak ada ID Obat yang sesuai"<<endl;  
 }  
 **else**{  
 **if** (PAPK->daftarObat->next == NULL){  
 //jika terdapat satu obat  
 PAPK->daftarObat = NULL;  
 **delete** M;  
 cout << "Obat Berhasil Dihapus"<<endl;  
 }  
  
 **else**{ // Jika Terdapat Banyak Obat  
 **if** (M == PAPK->daftarObat){  
 //jika ada di awal list  
 addressObat Q = PAPK->daftarObat;  
 **while** (Q->next != M){  
 Q = Q->next;  
 }  
 Q->next = NULL;  
 **delete** M;  
 cout<<"Obat Berhasil Dihapus"<<endl;  
 }  
 **else if** (M->next == NULL) {  
 addressObat Q = PAPK->daftarObat;  
 //jika berada di tenggah  
 **while** (Q->next != M){  
 Q = Q->next;  
 }  
 Q->next = M->next;  
 M->next = NULL;  
 **delete** M;  
 cout <<"Obat Berhasil Dihapus"<<endl;  
 }  
 **else** {  
 addressObat Q = PAPK->daftarObat;  
 //Jika Berada di akhir  
 **while** (Q->next != M){  
 Q = Q->next;  
 }  
 Q->next = NULL;  
 **delete** M;  
 cout << "Obat Berhasil Dihapus"<<endl;  
  
 }  
 }  
 }  
 }  
 }  
}

Program meminta user untuk menginputkan Id elemen parent jika elemen terdapat dalam list maka program akan mengoutputkan obat-obat pada apotek yang di cari. Kemudian program meminta user menginputkan id dari elemen obat yang ingin di hapus. jika elemen berada pada urutan pertama di list memanggil delete first, jika elemen berada di urutan terakhir program memanggil delete last dan jika tidak keduanya maka program menghapus elemen dengan menggunakan konsep delete after.

Prosedure Menu Delete Child Obat

**void** deleteObatOption(apotek &APK){  
 **if** (APK.first == NULL){  
 cout<<"Tidak Terdapat Apotek"<<endl;  
 }  
 **else** {  
 **char** menu;  
 ObatAwal :  
 cout << "Jenis Metode Delete Yang Ingin Dilakukan" << endl  
 << "1. Pilih Delete First" << endl  
 << "2. Pilih Delete Last" << endl  
 << "3. Pilih Delete After" << endl  
 << "4. Pilih Delete By ID" << endl  
 << "5. Exit" << endl;  
  
 cout << "Masukkan Pilihan : ";  
 cin >> menu; cin.ignore();  
 **switch** (menu) {  
 **case** '1' :  
 deleteObatFirst(APK);  
 cout << "Delete Data APK Berhasil" << endl; **break**; **goto** ObatAwal;  
 **case** '2' :  
 deleteObatLast(APK);  
 cout << "Delete Data APK Berhasil" << endl; **break**; **goto** ObatAwal;  
 **case** '3' :  
 deleteAfterAPK(APK);  
 cout << "Delete Data APK Berhasil" << endl; **break**; **goto** ObatAwal;  
 **case** '4' :  
 deleteObatByID(APK);  
 cout << "Delete Data APK Berhasil" << endl; **break**; **goto** ObatAwal;  
 **case** '5' : **goto** ObatAkhir;  
 **default** : cin.ignore(); **goto** ObatAwal;  
 }  
 ObatAkhir :;  
 }  
}

Program mengoutputkan menu tipe-tipe delete yang bisa dilakukan dan program meminta user menginputkan delete yang mana yang di inputkan, untuk menghapus child.

**Prosedure Insert First APK**

**void** insertFirstAPK(apotek &R, addressAPK P){  
 **if** (first(R)== NULL){//kondisi ketika list kosong  
 first(R) = P;  
 last(R) = P;  
 }  
 **else** {  
 P->next = R.first;  
 R.first->prev =P;  
 R.first = P;  
 }  
}

Meletakkan elemen pada urutan pertama di list parent, jika list kosong maka first dan last dari list tersebut langsung menunjuk elemen yang dimasukkan. Jika tidak maka element first akan menjadi next dari elemen yang di inputkan dan first berubah menjadi elemen yang di inputkan.

**Prosedure Insert Last APK**

**void** insertLastAPK(apotek &R, addressAPK P){  
 **if**(first(R) == NULL){  
 //kondisi ketika list ksosong  
 first(R) = P;  
 last(R) = P;  
 }  
 **else** {  
 prev(P) = last(R);  
 next(prev(P)) = P;  
 last(R) = P;  
 }  
}

Meletakkan elemen pada urutan terakhir di list parent, jika list kosong maka first dan last dari list tersebut langsung menunjuk elemen yang dimasukkan. Jika tidak maka element last akan menjadi prev dari elemen yang di inputkan dan last berubah menjadi elemen yang di inputkan.

**Prosedure Insert After APK**

**void** insertAfterAPK(apotek &R, addressAPK P){  
 **if** (R.first == NULL){  
 cout<<"karena Element Kosong maka Element langsung di insert di awal list"<<endl;  
 insertFirstAPK(R, P);  
 }  
 **else** {  
 string SearchByID;  
 printInfoAPK(R);  
 cout<<endl;  
 cout<<"Masukkan ID APK yang ingin di Insert Sesudahnya: ";getline(cin, SearchByID);  
 addressAPK PAPK;  
 PAPK = findElmAPK(R, SearchByID);  
 **if** (PAPK == NULL){  
 cout<<"element yang akan di Insert Sesudahnya Tidak Tersedia"<<endl;  
 }  
 **else** {  
 **if** (PAPK->next == NULL){  
 insertLastAPK(R, P);  
 }  
 **else** {  
 P->next = PAPK->next;  
 P->prev = PAPK;  
 PAPK->next->prev = P;  
 PAPK->next = P;  
 }  
 }  
 }  
}

Program akan meletakkan elemen yang di inputkan pada parameter setelah elemen yang di inputkan id-nya, jika list kosong akan memanggil insert first, jika elemen yang di inputkan id nya berada pada last maka program memanggil insert last, jika tidak maka elemen di insertkan menggunakan konsep insert after.

**Prosedure Menu Insert APK**

**void** addapotek(apotek &R){  
 infotypeAPK addAPK;  
 createAPK(addAPK);  
 addressAPK PAPK;  
 PAPK = alokasiAPK(addAPK);  
 **char** menu;  
 cout<<endl;  
 cout<<"Pilih Jenis Insert : "<<endl  
 <<"1. Insert First" <<endl  
 <<"2. Insert Last" <<endl  
 <<"3. Insert After" <<endl;  
 ObatLabel :  
 cout<<"Pilihan : "<<endl; cin>>menu; cin.ignore();  
 **switch** (menu){  
 **case** '1' : insertFirstAPK(R, PAPK); **break**;  
 **case** '2' : insertLastAPK(R, PAPK); **break**;  
 **case** '3' : insertAfterAPK(R, PAPK); **break**;  
 **default** : cin.ignore(); **goto** ObatLabel;  
 }  
 cout <<"\nAlhamdulillah Data Berhasil Di Insert"<<endl;  
}

Program mengoutputkan menu tipe-tipe insert yang bisa dilakukan dan program meminta user menginputkan insert yang mana yang di inputkan, untuk di insertkan sebagai apotek (list parent).

**Prosedure Count Total APK**

**void** countTotalAPK(apotek APK){  
 addressAPK PAPK;  
 PAPK = APK.first;  
 **if** (PAPK == NULL){  
 cout<<"Total Apotek yang Terdata Tidak Ada Bro"<<endl;  
 }  
 **else** {  
 **int** countNum = 0;  
 **while** (PAPK != NULL) {  
 countNum++;  
 PAPK = PAPK->next;  
 }  
 cout << "Total APotek yang Tersedia Sekarang Adalah : " << countNum << "Apotek" << endl;  
 }  
}

Mengoutputkan jumlah dari elemen parent (apotek) pada list parent.

**Prosedure Count Total Obat**

**void** countTotalObat(apotek APK){  
 **if** (APK.first == NULL)  
 cout<<"Tidak ada Data Obat"<<endl;  
 **else** {  
 string SearchByIDAPK;  
 string SearchByIDObat;  
 printInfoAPK(APK);  
 cout<<endl;  
 cout<<"Masukkan ID Apotek yang akan dihitung Jumalah Jenis Obatnya : "; getline(cin, SearchByIDAPK);  
 addressAPK PAPK = findElmAPK(APK, SearchByIDAPK);  
 **if** (PAPK != NULL){  
 **int** countNum = 0; //counter hitung  
 addressObat M = PAPK->daftarObat;  
 //looping hitung element  
 **while** (M != NULL){  
 countNum++;  
 M = M->next;  
 }  
 cout<<endl;  
 cout<<"Apotek : "<<PAPK->info.namaAPK<<" - Mempunyai "<<countNum<<" Obat"<<endl;  
 }  
 **else** {  
 cout<<endl;  
 cout<<"Tidak Terdapat ID Apotek"<<endl;  
 }  
 }  
}

Program memintainput Id dari elemen parent kemudian mengoutputkan jumlah dari elemen child (obat) pada list obat di parent tersebut.

**Prosedure Menu Count**

**void** menucountObat(apotek APK){  
 **char** menu;  
 cout<<"====== Pilih Element List Yang Mau Dihitung ======"<<endl  
 <<"1. Apotek"<<endl  
 <<"2.Obat Di Suatu Apotek"<<endl  
 <<"3. Kembali Ke Menu Utama"<<endl;  
 ObatLabel :  
 cout<<"Pilihan : ";cin>>menu; cin.ignore();  
 **switch** (menu){  
 **case** '1' : countTotalAPK(APK); **break**;  
 **case** '2' : countTotalObat(APK); **break**;  
 **case** '3' : **break**;  
 **default** : **goto** ObatLabel;  
 }  
}

Program mengoutputkan menu tipe-tipe count yang bisa dilakukan dan program meminta user menginputkan count yang mana yang di inputkan, untuk di counter jumlahnya.

**Prosedure Sorting**

**int** countLengthObat(addressObat PAPK){  
 **int** countNum = 0;  
 **if**(PAPK == NULL){  
 **return** 0;  
 } **else** {  
 addressObat M;  
 M = PAPK;  
 **while**(M != NULL){  
 countNum++;  
 M = M->next;  
 }  
 **return** countNum;  
 }  
}  
  
addressObat deleteObatFirstMergeSort(addressObat &M){  
   
 addressObat S, Q;  
 S = **new** elmObat;  
 S->info = M->info;  
 S->next = NULL;  
 Q = M;  
  
 **if** (M == NULL) {  
 cout << "Tidak Terdapat Obat"<<endl;  
 }  
 **else** {  
 //cout<<M->next<<endl;  
 **if** (M->next == NULL)  
 {  
 M = NULL;  
 **return** S;  
 }  
 **else** {  
 //cout<<Q->next->info.namaObat<<endl;  
 M = Q->next;  
 Q->next = NULL;  
 **return** Q;  
 }  
 }  
}  
  
**void** insertObatFirstMergeSort(addressObat &PAPK, addressObat M) {  
 **if** (PAPK == NULL) {  
 PAPK = M;  
 }  
 **else** {  
 M->next = PAPK;  
 PAPK = M;  
 }  
}  
  
**void** insertObatLastMergeSort(addressObat &PAPK, addressObat P) {  
 addressObat Q = PAPK;  
 **if**(Q == NULL){  
 insertObatFirstMergeSort(PAPK,P);  
 } **else** {  
 **while**(Q->next != NULL){  
 Q = Q->next;  
 }  
 Q->next = P;  
 }  
}  
  
addressObat mergeInObat(addressObat left, addressObat right){  
 addressObat resultList;  
 resultList = NULL; //List Baru untuk Penampung Element Terurut  
 addressObat M; //Address Buat Menampung data Pop untuk diinsert  
  
 **while**((left != NULL) && (right != NULL)){ //Selama Salah Satu List Belum Habis  
 **if**(left->info.idObat <= right->info.idObat){  
 M = deleteObatFirstMergeSort(left);  
 } **else** {  
 M = deleteObatFirstMergeSort(right);  
 }  
 insertObatLastMergeSort(resultList,M); //Memasukan Element yang di Pop dari Salah Satu List, (Right/Left)  
 //cout<<"Z"<<endl;  
 }  
  
 // Jika terdapat sisa Element dari List dari Kiri / Kanan (Sisanya pasti terurut karena proses rekursif awal)  
 **while**(left != NULL){  
 M = deleteObatFirstMergeSort(left);  
 insertObatLastMergeSort(resultList,M);  
 //cout<<"y"<<endl;  
 }  
 **while**(right != NULL){  
 M = deleteObatFirstMergeSort(right);  
 insertObatLastMergeSort(resultList,M);  
 //cout<<"x"<<endl;  
 }  
 **return** resultList;  
}  
  
addressObat mergeSortObat(addressObat PAPK){  
 **int** lengthList = countLengthObat(PAPK);  
 //cout<<lengthList<<endl;  
 addressObat tmpList; // Ini nantinya akan jadi representatif list Kiri  
 tmpList = NULL;  
 addressObat M;  
  
 // Jika List Elemenya Satu Return List Tersebut  
 **if**(lengthList <=1){  
 **return** PAPK;  
 }  
 //  
 **int** i = 0;  
  
  
 **while**(i < lengthList/2){  
 M = deleteObatFirstMergeSort(PAPK);  
 insertObatFirstMergeSort(tmpList,M);  
 i++;  
   
 }  
 // Recursively sort both sublists  
 tmpList = mergeSortObat(tmpList);  
 //cout<<"X"<<endl;  
 PAPK = mergeSortObat(PAPK);  
  
 **return** mergeInObat(tmpList,PAPK);  
}  
  
**void** sortObat(apotek &APK){  
 **if** (APK.first == NULL)  
 {  
 cout << "Tidak Ada Apotek"<<endl;  
 }  
 **else** {  
 string strSearchID;  
 printInfoAPK(APK);  
 cout << endl;  
 cout << "Masukkan ID Apotek yang akan diSorting Obatnya: ";getline(cin, strSearchID);  
 addressAPK PAPK = findElmAPK(APK, strSearchID);  
 //cout<<PAPK->daftarObat->info.namaObat<<endl;  
 **if** (PAPK == NULL)  
 {  
 cout << "Tidak Ada ID Apotek Yang Sesuai"<<endl;  
 }  
 **else** {  
 **if** (PAPK->daftarObat == NULL)  
 {  
 cout << "Tidak Terdapat Obat"<<endl;  
 }  
 **else** {  
   
 PAPK->daftarObat = mergeSortObat(PAPK->daftarObat);  
 //cout<<"y"<<endl;  
 cout<<"========== Sorting pada "<<PAPK->info.namaAPK<<" Telah Berhasil Dilakukan =========="<<endl;  
 cout << endl;  
 cout << "HASIL :"<<endl;  
 cout << endl;  
 addressObat M = PAPK->daftarObat;  
 **while** (M != NULL)  
 {  
 cout <<"ID Obat : "<<M->info.idObat<<endl;  
 cout <<"Nama Obat : "<<M->info.namaObat<<endl;  
 cout<<"Jenis Obat : "<<M->info.jenisObat<<endl;  
 cout<<"Stock Obat : "<<M->info.stock<<endl;  
 cout<<"Kadaluarsa : "<<M->info.kadaluarsa<<endl;  
 cout <<"Harga Obat : "<<M->info.hargaObat<<endl;  
 cout << endl;  
 M = M->next;  
 }  
 }  
 }  
 }  
}

Program Utama

Main.cpp

Inisalisasi Variabel

**int** main () {  
 apotek mainAPK;  
 createListAPK(mainAPK);  
 addressAPK addAPK;  
 infotypeAPK dataAPK;  
 addressObat addObat;  
 infotypeObat dataObat;  
  
 {

Memasukkan elemen secara hardcode

dataAPK.idAPK = "APK-0001";  
dataAPK.namaAPK = "Apotek Setia Budi";  
dataAPK.alamat= "Jalan Bismillah Bandung";  
dataAPK.tanggal=" " ;  
addAPK = alokasiAPK(dataAPK);  
insertLastAPK(mainAPK, addAPK);  
  
  
dataAPK.idAPK = "APK-0002";  
dataAPK.namaAPK = "Apotek Hamasah";  
dataAPK.alamat= "Jalan Kebenaran Bandung";  
dataAPK.tanggal=" " ;  
addAPK = alokasiAPK(dataAPK);  
insertLastAPK(mainAPK, addAPK);  
  
  
  
dataAPK.idAPK = "APK-0003";  
dataAPK.namaAPK = "Apotek Cinta";  
dataAPK.alamat= "Jalan Putus Nyambung Bandung";  
dataAPK.tanggal=" " ;  
addAPK = alokasiAPK(dataAPK);  
insertLastAPK(mainAPK, addAPK);  
  
  
  
dataAPK.idAPK = "APK-0004";  
dataAPK.namaAPK = "Apotek Kimia Farma";  
dataAPK.alamat= "Jalan atom Bandung";  
dataAPK.tanggal=" " ;  
addAPK = alokasiAPK(dataAPK);  
insertLastAPK(mainAPK, addAPK);  
  
  
  
dataAPK.idAPK = "APK-0005";  
dataAPK.namaAPK = "Apotek Genesis";  
dataAPK.alamat= "Jalan Terminator Bandung";  
dataAPK.tanggal=" " ;  
addAPK = alokasiAPK(dataAPK);  
insertLastAPK(mainAPK, addAPK);  
  
//Test case Obat  
  
dataObat.idObat="MED-001";  
dataObat.namaObat="Xamarin";  
dataObat.jenisObat="Migran";  
dataObat.hargaObat="10000";  
dataObat.stock= "17";  
dataObat.kadaluarsa="15 September 2017";  
addAPK = findElmAPK(mainAPK, "APK-0001");  
insertObatFirst(addAPK, dataObat);  
  
  
dataObat.idObat="MED-002";  
dataObat.namaObat="Maltos";  
dataObat.jenisObat="pusing";  
dataObat.hargaObat="20000";  
dataObat.stock="147";  
dataObat.kadaluarsa="15 Desember 2017";  
addAPK = findElmAPK(mainAPK, "APK-0002");  
insertObatFirst(addAPK, dataObat);  
  
  
dataObat.idObat="MED-002";  
dataObat.namaObat="Panasilin";  
dataObat.jenisObat="luka-luka";  
dataObat.hargaObat="80000";  
dataObat.stock="117";  
dataObat.kadaluarsa="15 Oktober 2017";  
addAPK = findElmAPK(mainAPK, "APK-0003");  
insertObatFirst(addAPK, dataObat);

}

Menu Program Apotekpedia

ObatLabel:  
 system("cls");  
 cout << "#### TUGAS BESAR STRUKTUR DATA ####"<<endl  
 << " #### DEEP WEB TEAM ####"<<endl  
 <<endl  
 << " 1. Insert Toko Apotek"<<endl  
 << " 2. Insert Obat Apotek"<<endl  
 << " 3. Edit Info Apotek"<<endl // Berdasarkan ID  
 << " 4. Edit Info Obat Apotek"<<endl // Berdasarkan ID Obat  
 << " 5. Delete Apotek"<<endl // Berdasarkan ID  
 << " 6. Delete Obat APK"<<endl // Berdasarkan ID Obat  
 << " 7. Print Info Apotek"<<endl  
 << " 8. Print APK & Obatnya"<<endl  
 << " 9. Search Apotek"<<endl //Berdasarkan ID APK  
 << " 10. Search Obat APK"<<endl //Berdasarkan Apotek  
 << " 11. Sorting Apotek by ID"<<endl  
 << " 12. Sorting Obat APK by ID"<<endl  
 << " 13. Hitung Total APK / Obat"<<endl  
 << " 0. EXIT"<<endl;  
 cout << endl;  
 cout << "Pilih Menu (Angka): "; **int** Obat;  
 cin>>Obat; cin.ignore();  
 **switch** (Obat) {  
 **case** 1 : system("cls"); addapotek(mainAPK); getch(); **goto** ObatLabel;  
 **case** 2 : system("cls"); addObatAPK(mainAPK); getch(); **goto** ObatLabel;  
 **case** 3 : system("cls"); editInfoAPK(mainAPK); getch(); **goto** ObatLabel;  
 **case** 4 : system("cls"); editObat(mainAPK); getch(); **goto** ObatLabel;  
 **case** 5 : system("cls"); deleteAPKOption(mainAPK); getch(); **goto** ObatLabel;  
 **case** 6 : system("cls"); deleteObatOption(mainAPK); getch(); **goto** ObatLabel;  
 **case** 7 : system("cls"); printInfoAPK(mainAPK); getch(); **goto** ObatLabel;  
 **case** 8 : system("cls"); printInfoObat(mainAPK); getch(); **goto** ObatLabel;  
 **case** 9 : system("cls"); searchAPK(mainAPK); getch(); **goto** ObatLabel;  
 **case** 10 : system("cls"); searchObat(mainAPK); getch(); **goto** ObatLabel;  
// case 11 : system("cls"); sortapotek(mainAPK); getch(); goto ObatLabel;  
 // case 12 : system("cls"); sortObat(mainAPK); getch(); goto ObatLabel;  
 **case** 13 : system("cls"); menucountObat(mainAPK); getch(); **goto** ObatLabel;  
 **case** 0 : **goto** ExitLabel; // Case Exit  
 **default** : cin.ignore(); **goto** ObatLabel; // Case Handling Inputan Lain  
 }  
 ExitLabel: ;  
  
 **return** 0;  
}

# Output Program

|  |  |
| --- | --- |
| Gambar 1 Menu Utama Program | Gambar 2 Mencari Id Apotek |
| Gambar 3 Sorting By ID APK | Gambar 4 Count Total APK |
| Gambar 5 Insert Obat ke parent | Gambar 6 Insert Parent APK |

|  |  |
| --- | --- |
| Gambar 7 Edit APK | Gambar 8 Edit Obat di Parent |
| Gambar 9 Delete Apotek | Gambar 10 Print Info APK dan Obatnya |