MAKALAH

PEMROGRAMAN TERSTRUKTUR

TUGAS BESAR PEMBUATAN PROGRAM PENDAFTARAN PON 2016

****

**DISUSUN OLEH:**

1. DWI WAHYU AKBAR SADEWO (1302140092)
2. RADIKA RAFIF GIBRAN (1302140152)
3. RIYAN NURSYALIM (1302140142)

IK-38-02

**TELKOM UNIVERSITY**

Jalan Telekomunikasi - Terusan Buahbatu

Bandung, Jawa Barat 40257

Indonesia

1. **Latar Belakang**

Latar belakang kami untuk membuat program ini adalah untuk mempermudah para panitia dan peserta dalam melakukan registrasi, PON 2016 diikuti kurang lebih 33 provinsi se-Indonesia. Dan juga cabang perlombaan yang diikuti tidak hanya satu, melainkan banyak cabang yang diikuti.

Dengan latar belakang inilah kami berinisiatif untuk membuat program ini. Sehingga bisa membantu para panitia dan peserta PON 2016 untuk mempermudah melakukan registrasi peserta dan sangat efisien dalam hal waktu, tidak perlu membuang buang kertas untuk hal yang tidak perlu. Dengan program ini saja kita sudah menghemat milliaran rupiah uang negara untuk membeli kertas registrasi.

**1.2 Rumusan masalah**

1. Bagaimana penggunaan algoritma dalam kehidupan sehari-hari ?
2. Apa tujuan dari pembuatan program ini?
3. Bagaimana gambaran sistem kerja program ini ?

**1.3 Maksud dan Tujuan**

1. Untuk mengetahui apa saja kegunaan algoritma dalam kehidupan kita
2. Untuk mengetahui apa tujuan dari program ini
3. Untuk mengetahui bagaimana gambaran sistem kerja program ini

DESKRIPSI SOFTWARE

Pada deskripsi software ini kita akan membahas tentang kendala yang kami temukan selama pembuatan program ini. Dimana program ini kami buat dalam rangka memenuhi tugas besar Pemrograman Terstruktur II. Berikut akan kami paparkan penjelasan mengenai kendala yang kami dapat.

1. Penggunaan algoritma dalam kehidupan sehari-hari

Di awal tadi sudah dijelaskan mengenai apa itu algoritma, dimana intinya algoritma adalah urutan langkah - langkah logis untuk penyelesaian masalah yang disusun secara sistematis. Adapun penggunaan algoritma dalam kehidupan sehari-hari

1. Tujuan dari pembuatan program ini

Adapun tujuan dari program ini adalah untuk membantu mempermudah seseorang dalam mengatur pendaftaran peserta PON 2016

1. Gambaran sistem kerja program

Dalam program ini kami menggunakan multilist. Dapat kami gambarkan dalam flowchart berikut.

FLOWCHART Program pon 2016

Mulai

Mengisi nama peserta, lomba yang diikuti, dan asal daerah.

Logoutut

Selesai

Selesai

Melihat Data Peserta

Input Nama Peserta

Exit

Benar

Prosedur

1.Tambah Peserta

2.Tambah Lomba

3.Hapus Peserta

4.Hapus Daerah

5.Lihat Data Peserta

6.Lihat daftar Lomba

7.Lihat data Kategori

8.Lihat daftar daerah

Username & Password

End

Login Sebagai Panitia

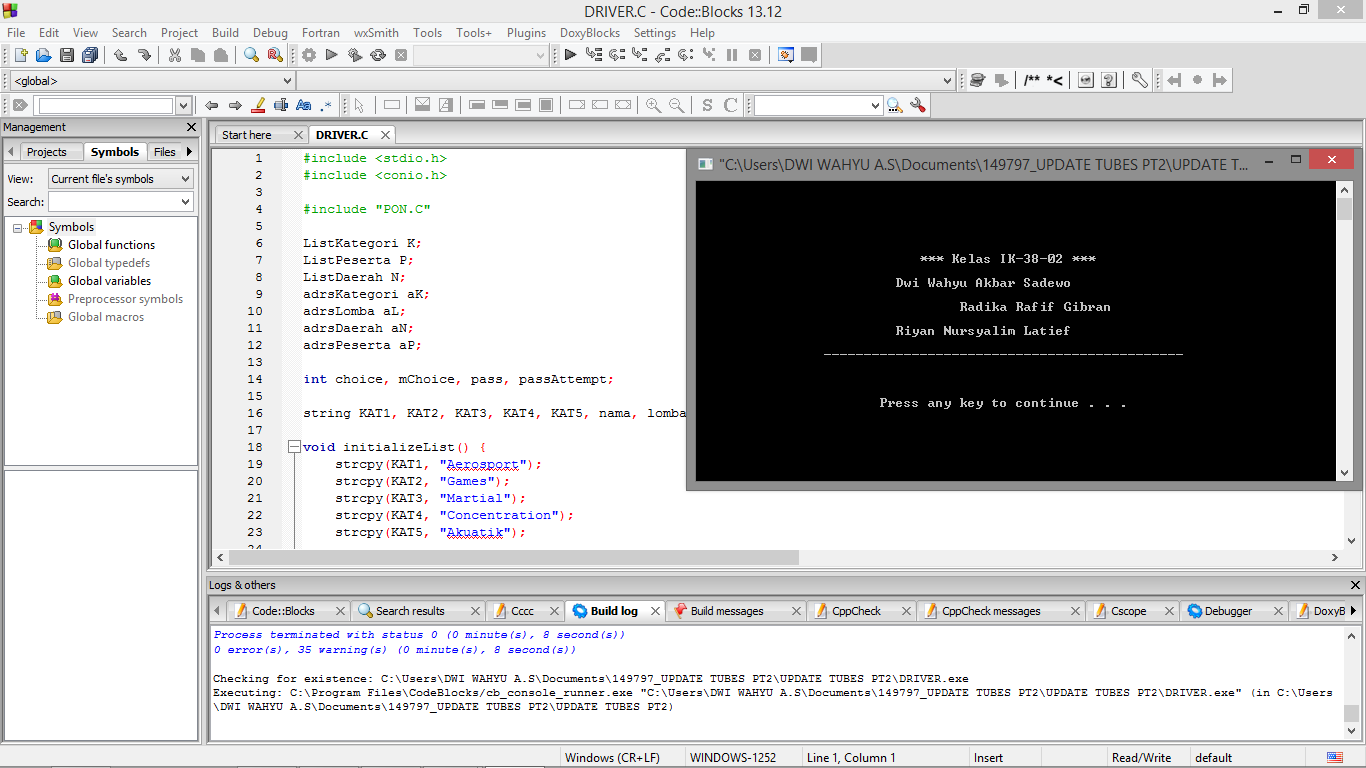
Lihat Data Peserta

Daftar Sebagai Peserta

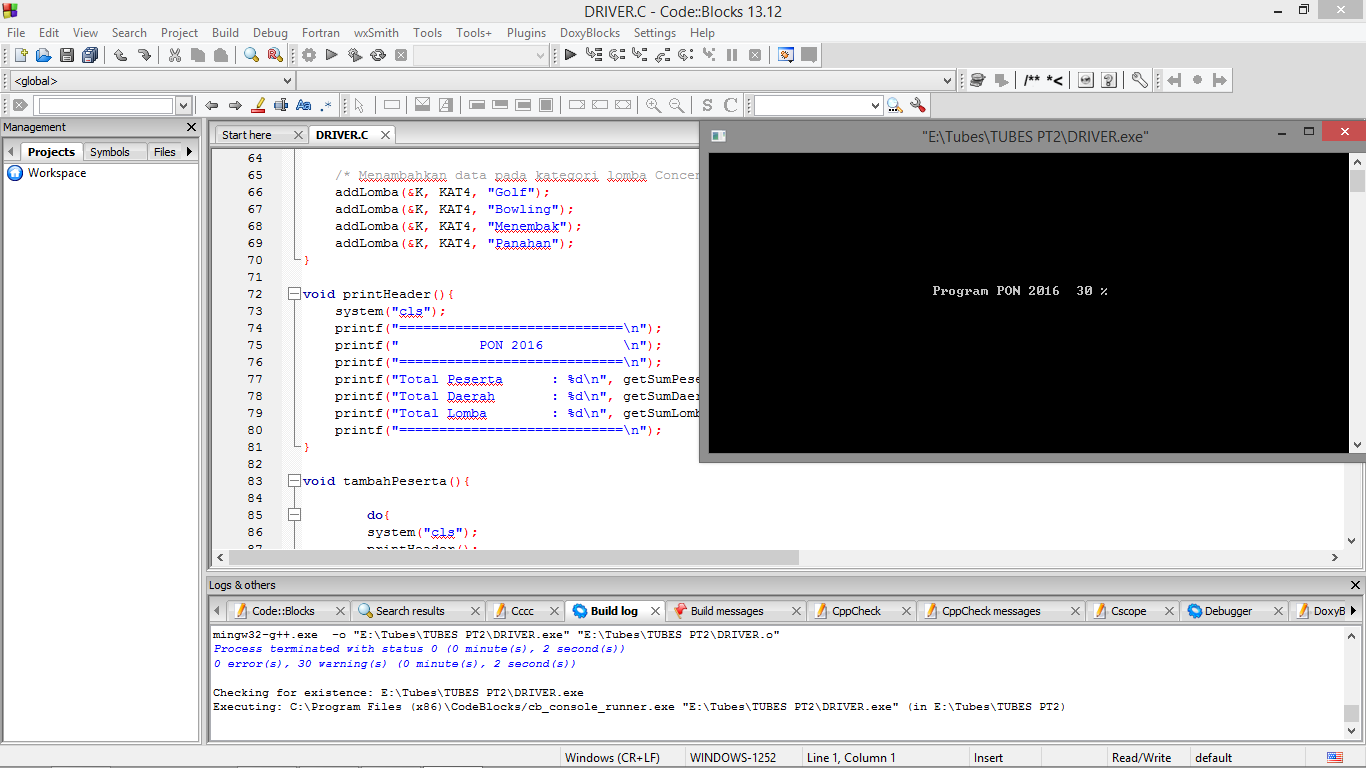
Input Pilihan

Menu Utama

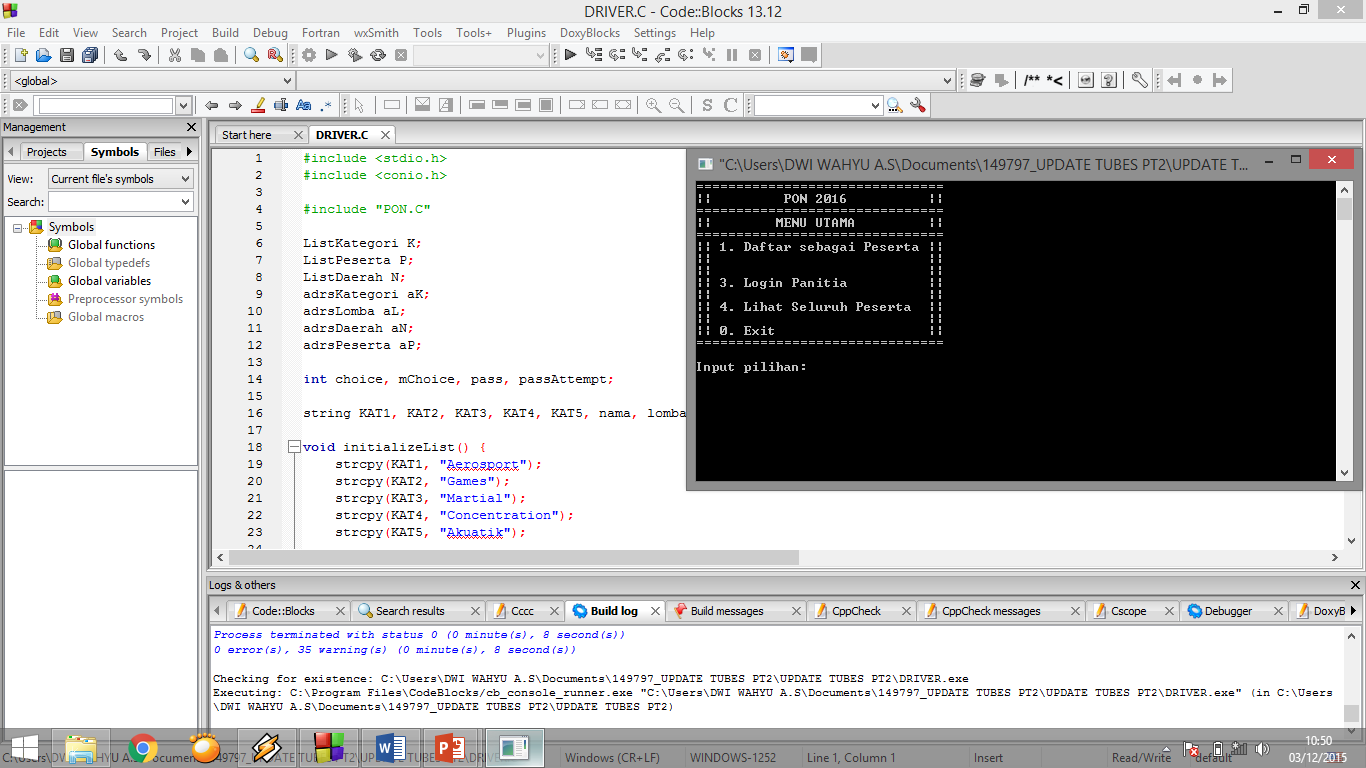
SCREENSHOOT PROGRAM



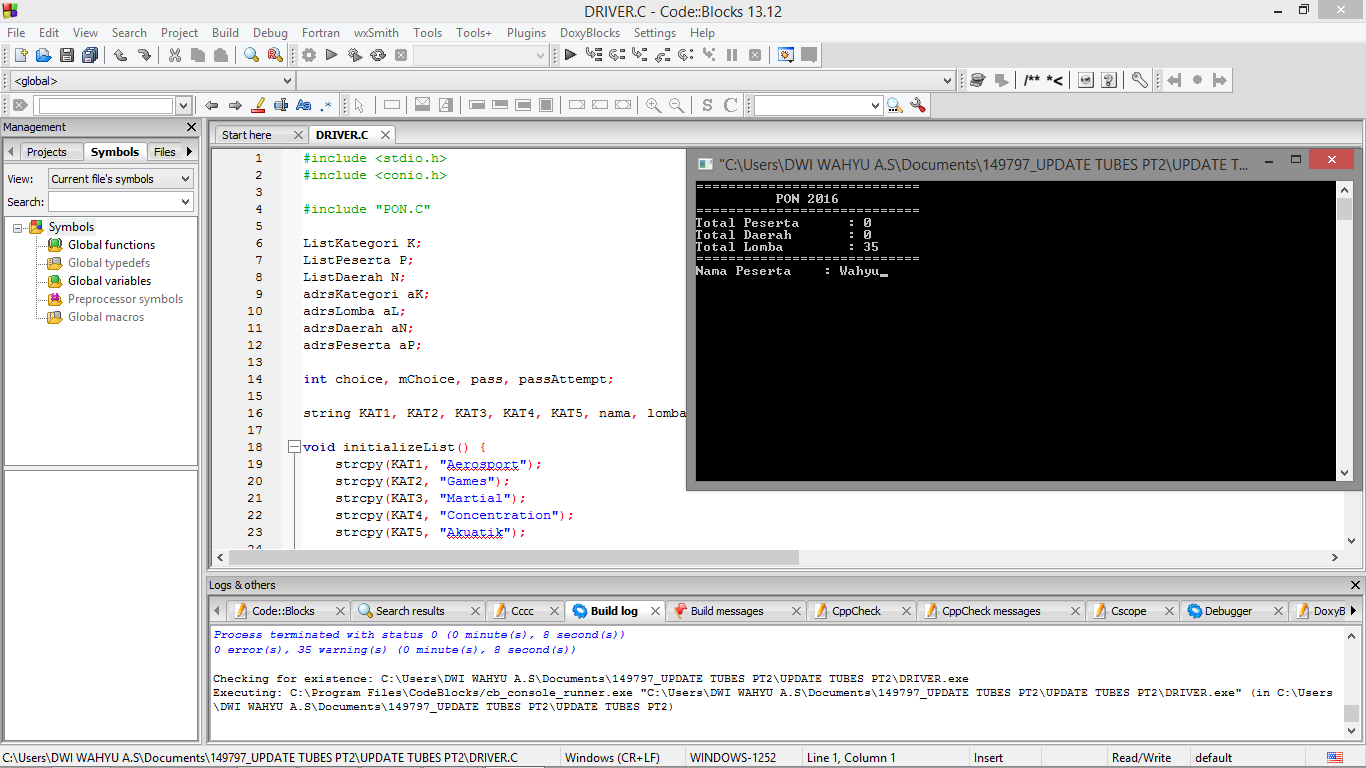
Tampilan Awal Program PON 2016



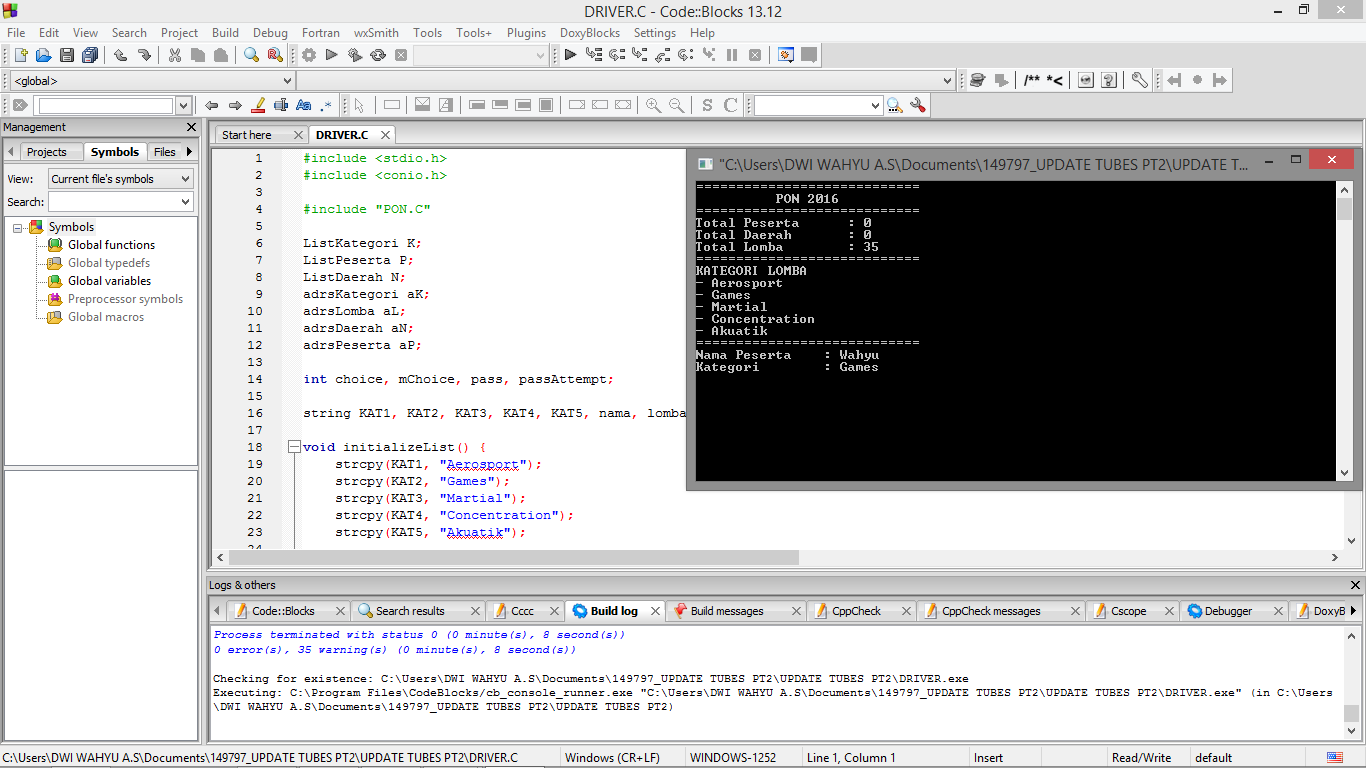
Tampilan Kedua Program PON 2016



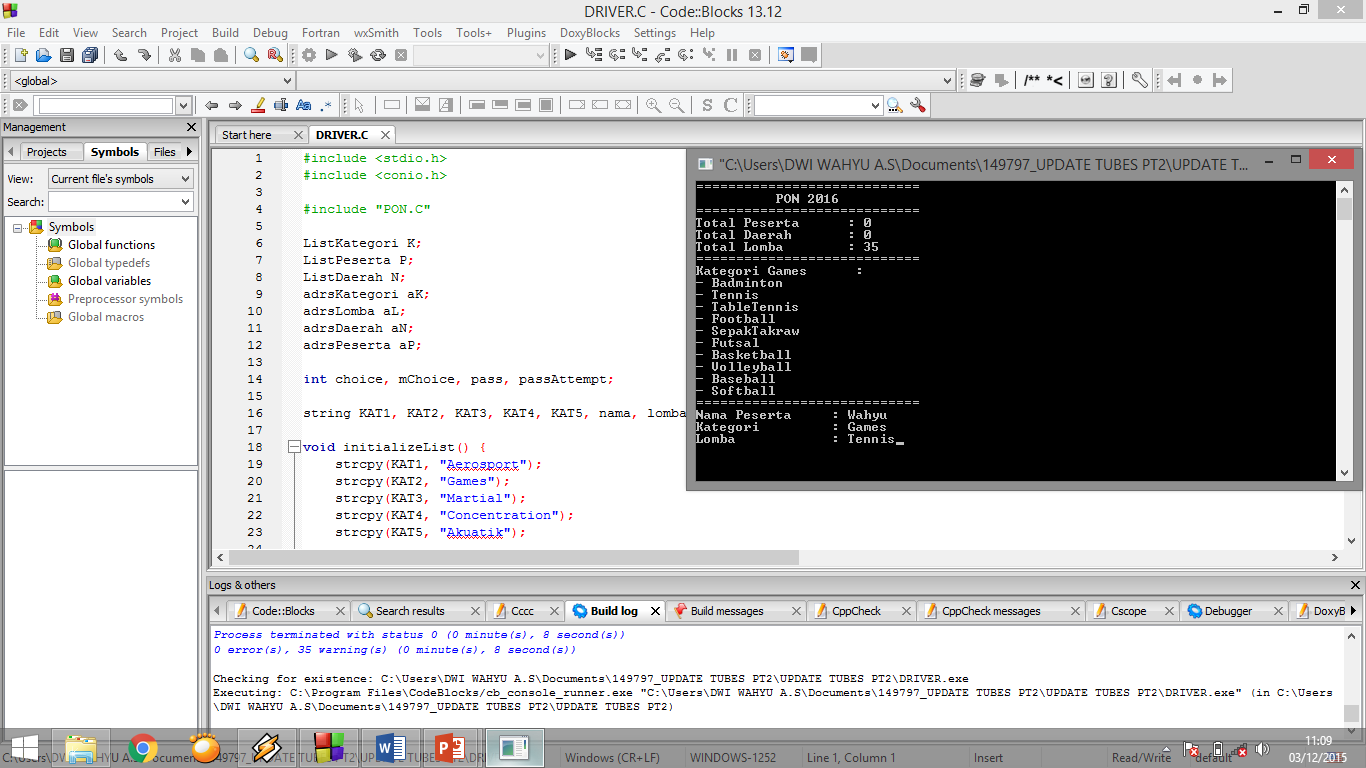
Tampilan Menu Utama Program PON 2016



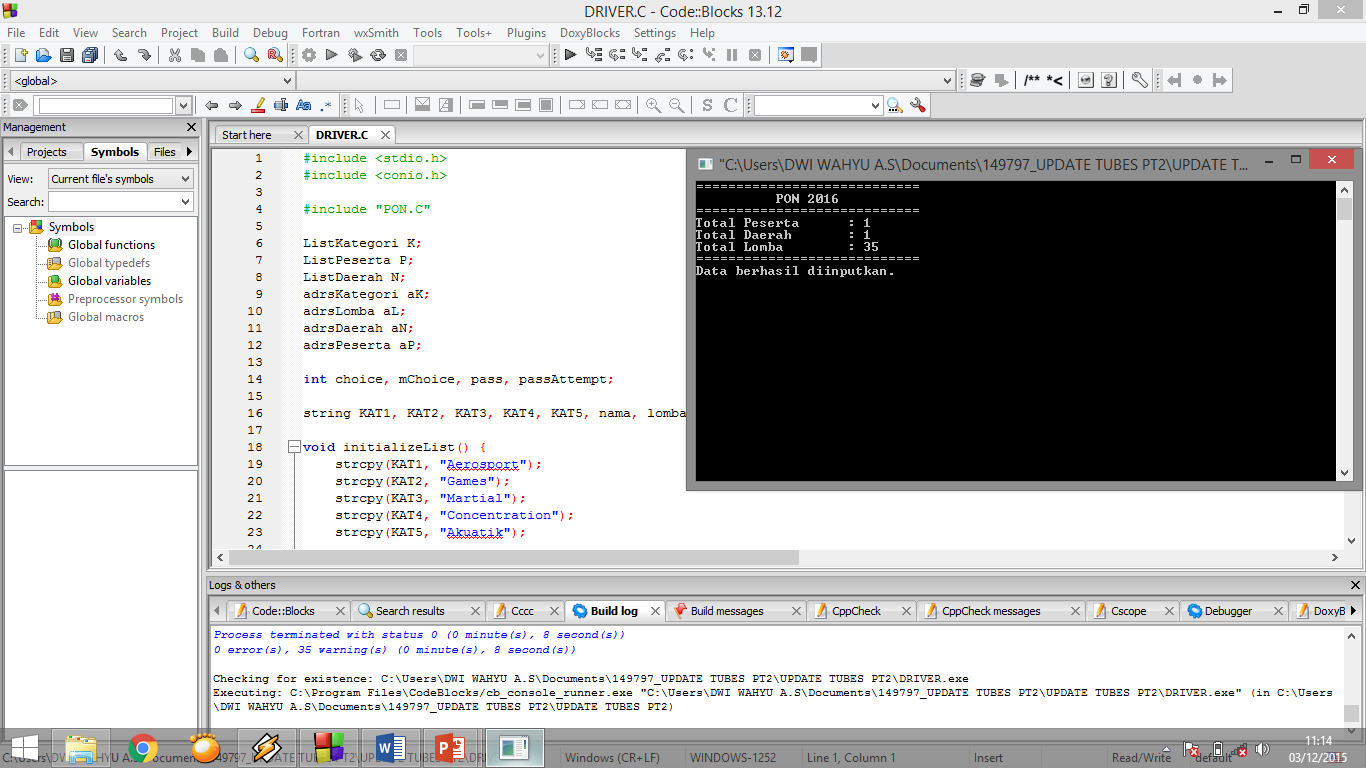
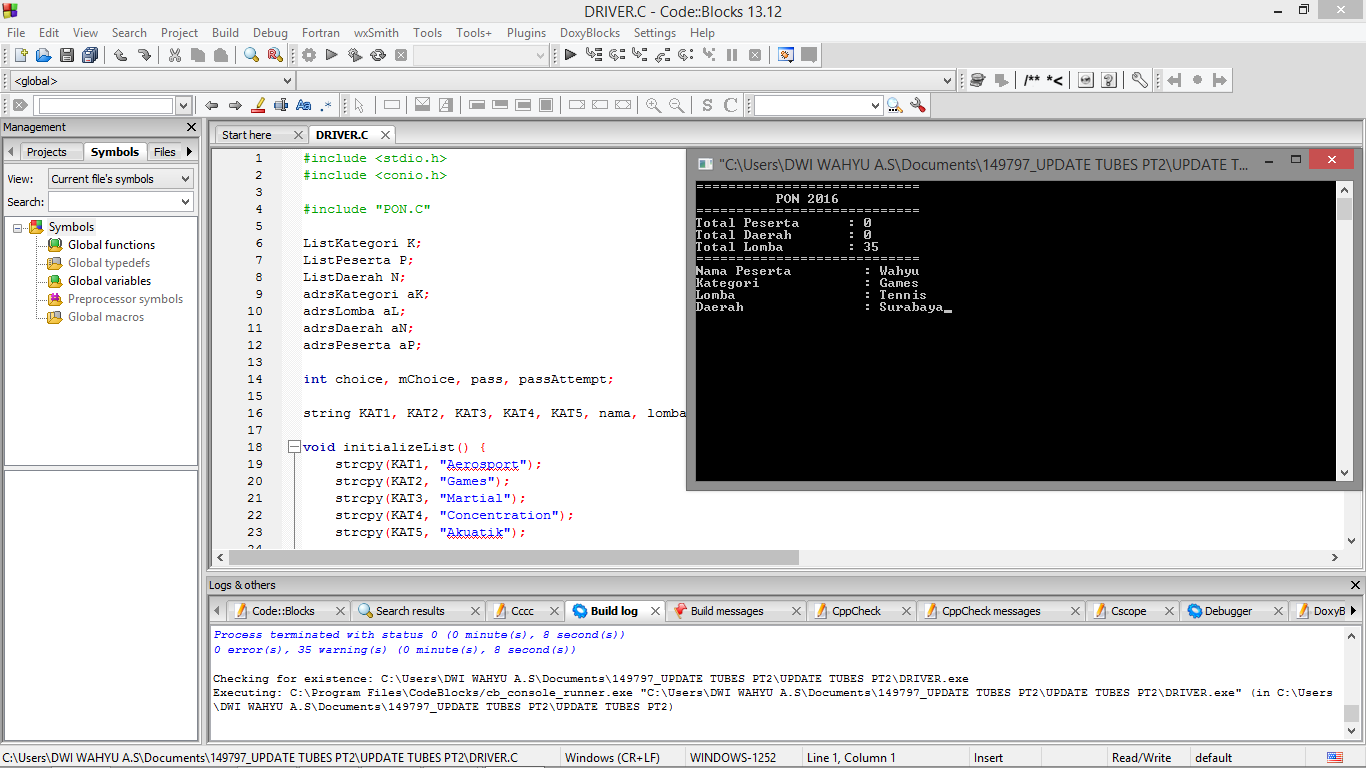
Peserta diminta memasukkan Nama Peserta



Peserta diminta memasukkan Kategori Lomba yang diikuti



Peserta diminta memasukkan Jenis Lomba yang diikuti



Peserta diminta memasukkan dari Daerah dia berasal

Data Peserta berhasil diinputkan.

PSEUDECODE

* **DRIVER .C**

#include <stdio.h>

#include <conio.h>

#include "PON.C"

ListKategori K;

ListPeserta P;

ListDaerah N;

adrsKategori aK;

adrsLomba aL;

adrsDaerah aN;

adrsPeserta aP;

int choice, mChoice, pass, passAttempt;

string KAT1, KAT2, KAT3, KAT4, KAT5, nama, lomba, daerah, kategori, tmp;

void initializeList() {

strcpy(KAT1, "Aerosport");

strcpy(KAT2, "Games");

strcpy(KAT3, "Martial");

strcpy(KAT4, "Concentration");

strcpy(KAT5, "Akuatik");

createListKategori(&K);

createListPeserta(&P);

createListDaerah(&N);

addKategori(&K, KAT1);

addKategori(&K, KAT2);

addKategori(&K, KAT3);

addKategori(&K, KAT4);

addKategori(&K, KAT5);

/\* Menambahkan data pada kategori lomba Aerosports \*/

addLomba(&K, KAT1, "Aeromodelling");

addLomba(&K, KAT1, "Gantolle");

addLomba(&K, KAT1, "Paralayang");

addLomba(&K, KAT1, "TerbangLayang");

addLomba(&K, KAT1, "TerjunPayung");

/\* Menambahkan data pada kategori lomba Games Sports \*/

addLomba(&K, KAT2, "Badminton");

addLomba(&K, KAT2, "Tennis");

addLomba(&K, KAT2, "TableTennis");

addLomba(&K, KAT2, "Football");

addLomba(&K, KAT2, "SepakTakraw");

addLomba(&K, KAT2, "Futsal");

addLomba(&K, KAT2, "Basketball");

addLomba(&K, KAT2, "Volleyball");

addLomba(&K, KAT2, "Baseball");

addLomba(&K, KAT2, "Softball");

/\* Menambahkan data pada kategori lomba Martial Art \*/

addLomba(&K, KAT3, "Wushu");

addLomba(&K, KAT3, "Silat");

addLomba(&K, KAT3, "Karate");

addLomba(&K, KAT3, "Taekwondo");

addLomba(&K, KAT3, "Silat");

addLomba(&K, KAT3, "Judo");

addLomba(&K, KAT3, "Gulat");

addLomba(&K, KAT3, "Tinju");

addLomba(&K, KAT3, "Binaraga");

/\* Menambahkan data pada kategori lomba Concentration Sports \*/

addLomba(&K, KAT4, "Golf");

addLomba(&K, KAT4, "Bowling");

addLomba(&K, KAT4, "Menembak");

addLomba(&K, KAT4, "Panahan");

addLomba(&K, KAT4, "Billiard");

addLomba(&K, KAT4, "Catur");

/\* Menambahkan data pada kategori lomba Akuatik Sports \*/

addLomba(&K, KAT5, "Renang");

addLomba(&K, KAT5, "RenangIndah");

addLomba(&K, KAT5, "LoncatIndah");

addLomba(&K, KAT5, "PoloAir");

addLomba(&K, KAT5, "Dayung");

}

void printHeader(){

system("cls");

printf("============================\n");

printf(" PON 2016 \n");

printf("============================\n");

printf("Total Peserta : %d\n", getSumPeserta(P));

printf("Total Daerah : %d\n", getSumDaerah(N));

printf("Total Lomba : %d\n", getSumLomba(K));

printf("============================\n");

}

void tambahPeserta(){

do{

system("cls");

printHeader();

printf("Nama Peserta : ");

scanf("%s", &nama);

if (checkPesertaExistence(P, nama) == 1){

printf("Nama tersebut telah terdaftar sebelumnya.");

getch();

}

} while (checkPesertaExistence(P, nama) == 1);

do{

system("cls");

printHeader();

printf("KATEGORI LOMBA\n");

printf("- Aerosport\n");

printf("- Games\n");

printf("- Martial\n");

printf("- Concentration\n");

printf("- Akuatik\n");

printf("============================\n");

printf("Nama Peserta : %s\n", nama);

printf("Kategori : ");

scanf("%s", &kategori);

if (checkKategoriExistence(K, kategori) == 0){

printf("Kategori yang Anda inputkan tidak ada.");

getch();

}

} while (checkKategoriExistence(K, kategori) == 0);

do{

system("cls");

printHeader();

printf("Kategori %s : \n", kategori);

printLombaInKategori(K, kategori, 0);

printf("============================\n");

printf("Nama Peserta : %s\n", nama);

printf("Kategori : %s\n", kategori);

printf("Lomba : ");

scanf("%s", &lomba);

if (checkLombaExistence(K, kategori, lomba) == 0){

printf("Lomba yang Anda inputkan tidak ada.");

getch();

}

} while (checkLombaExistence(K, kategori, lomba) == 0);

system("cls");

printHeader();

printf("Nama Peserta : %s\n", nama);

printf("Kategori : %s\n", kategori);

printf("Lomba : %s\n", lomba);

printf("Daerah : ");

scanf("%s", &daerah);

if (checkDaerahExistence(N, daerah) == 0){

addDaerah(&N, daerah);

}

aN = getAddressDaerah(N, daerah);

aL = getAddressLomba(K, kategori, lomba);

addPeserta(&P, nama, aL, aN);

system("cls");

printHeader();

printf("Data berhasil diinputkan.");

getch();

}

void tambahLomba(){

system("cls");

printHeader();

printf("============================\n");

printf(" KATEGORI LOMBA \n");

printf(" - Aerosport \n");

printf(" - Games \n");

printf(" - Martial \n");

printf(" - Concentration \n");

printf(" - Akuatik \n");

printf("============================\n");

printf("Kategori : ");

scanf("%s", &kategori);

if (checkKategoriExistence(K, kategori) == 0){

printf("Kategori yang Anda inputkan tidak ada.\n");

getch();

}else{

while(checkKategoriExistence(K, kategori)!= 0){

system("cls");

printHeader();

printLombaInKategori(K, kategori, 0);

printf("============================\n");

printf("Kategori : %s\n", kategori);

printf("Lomba : ");

scanf("%s", &lomba);

if (checkLombaExistence( K, kategori, lomba) == 1){

printf("Lomba yang Anda inputkan sudah ada.\n");

getch();

}else{

while (checkLombaExistence(K, kategori, lomba) != 1){

addLomba(&K, kategori, lomba);

system("cls");

printHeader();

printf("Data lomba berhasil diinputkan\n");

getch();

break;

}

break;

}

break;

}

}

}

void lihatPeserta(){

system("cls");

printHeader();

printf("DAFTAR PESERTA\n");

printAllPeserta(P, N, K, 1);

getch();

}

void hapusPeserta(){

system("cls");

printHeader();

printf("DAFTAR PESERTA\n");

printAllPeserta(P, N, K, 0);

printf("\n");

printf("============================\n");

printf("Nama Peserta : ");

scanf("%s", &nama);

deletePeserta(&P, nama, &tmp);

system("cls");

printHeader();

printAllPeserta(P, N, K, 0);

printf("============================\n");

printf("\"%s\" telah dihapus dari daftar peserta.", tmp);

getch();

}

void hapusDaerah(){

system("cls");

printHeader();

printf("DAFTAR Daerah\n");

printAllDaerah(N);

printf("\n");

printf("============================\n");

printf("Nama Daerah : ");

scanf("%s", &daerah);

deleteDaerah(&N, daerah, &tmp);

system("cls");

printHeader();

printAllDaerah(N);

printf("============================\n");

printf("\"%s\" telah dihapus dari daftar daerah.", tmp);

getch();

}

void lihatLomba(){

system("cls");

printHeader();

printf("DAFTAR LOMBA\n");

printAllLomba(K);

getch();

}

void lihatKategori(){

system("cls");

printHeader();

printf("DAFTAR KATEGORI\n");

printf("- Aerosport\n");

printf("- Games\n");

printf("- Martial\n");

printf("- Concentration\n");

printf("- Akuatik\n");

printf("============================\n");

printf("Kategori : ");scanf("%s", &kategori);

printHeader();

printLombaInKategori(K, kategori, 1);

getch();

}

void lihatDaerah(){

system("cls");

printHeader();

printf("DAFTAR DAERAH\n");

printAllDaerah(N);

getch();

}

void logout(){

choice = 0;

mChoice = 10;

}

void lihatPesertaIndiv(){

do{

system("cls");

printf("===========================\n");

printf("|| PON 2016 ||\n");

printf("===========================\n");

printf("Nama Peserta : ");

scanf("%s", &nama);

if (checkPesertaExistence(P, nama) == 0){

printf("Data peserta yang Anda inputkan tidak ada.\n");

getch();

}

} while (checkPesertaExistence(P, nama) == 0);

system("cls");

printf("===========================\n");

printf("|| PON 2016 ||\n");

printf("===========================\n");

printPesertaIndiv(K, N, P, nama);

getch();

}

void lihatSeluruhPeserta(){

system("cls");

printf("===========================\n");

printf("|| PON 2016 ||\n");

printf("===========================\n");

printHeader();

printf("\tDAFTAR PESERTA PON 2016\n");

printAllPeserta(P, N, K, 1);

getch();

}

void login(){

pass = 0;

passAttempt = 0;

while (pass != 310597 && passAttempt != 3){

system("cls");

printf("=====================\n");

printf("|| LOGIN ||\n");

printf("=====================\n");

printf("\n");

printf("Password: ");

scanf("%d", &pass);

passAttempt++;

if (passAttempt == 3){

printf("\n \n");

printf("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n");

printf("x x\n");

printf("x Maaf, Anda salah memasukkan password selama 3 kali. x\n");

printf("x x\n");

printf("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n");

getch();

}

}

if (pass == 310597){

while (choice != 0) {

system("cls");

mChoice = 0;

printHeader();

printf("MENU\n");

printf("1. Tambah peserta\n");

printf("2. Tambah lomba\n");

printf("3. Hapus peserta\n");

printf("4. Hapus daerah\n");

printf("\n");

printf("5. Lihat data peserta\n");

printf("6. Lihat daftar lomba\n");

printf("7. Lihat data kategori\n");

printf("8. Lihat daftar daerah\n");

printf("\n");

printf("9. Logout\n");

printf("0. Exit\n");

printf("\n");

printf("Inputkan pilihan Anda: ");

scanf("%d", &choice);

switch(choice){

case 1: tambahPeserta();

break;

case 2: tambahLomba();

break;

case 3: hapusPeserta();

break;

case 4: hapusDaerah();

break;

case 5: lihatPeserta();

break;

case 6: lihatLomba();

break;

case 7: lihatKategori();

break;

case 8: lihatDaerah();

break;

case 9: logout();

break;

case 0: break;

default: printf("Input yang Anda masukkan salah.");

}

}

}

}

int main() {

char a[]={"\n\n\n\n\n\n\t\t\t \*\*\* Kelas IK-38-02 \*\*\*\n\n\t\t\t Dwi Wahyu Akbar Sadewo \n\n\t\t\t\t Radika Rafif Gibran\n\n\t\t\t Riyan Nursyalim Latief\_"};

int i,k,l;

double j;

for(i=0;a[i]!='\_';i++){

printf("%c",a[i]);

for(j=0;j<=9990000;j++){

}

}

puts("\n\n\t\t---------------------------------------------\n\n\n");

printf("\t\t ");

system("pause");

for (k=1;k<=100;k++){

printf("\n\n\n\n\n\n\n\n\n\n\n Program PON 2016 ");

printf("%i %%",k);

system("cls");

}

system("cls");

choice = 10;

mChoice = 10;

initializeList();

while(mChoice != 0){

system("cls");

printf("===============================\n");

printf("|| PON 2016 ||\n");

printf("===============================\n");

printf("|| MENU UTAMA ||\n");

printf("===============================\n");

printf("|| 1. Daftar sebagai Peserta ||\n");

printf("|| ||\n");

if (isPesertaEmpty(P) == 0){

printf("|| 2. Lihat Data Anda ||\n");

}

printf("|| ||\n");

printf("|| 3. Login Panitia ||\n");

printf("|| ||\n");

printf("|| 4. Lihat Seluruh Peserta ||\n");

printf("|| ||\n");

printf("|| 0. Exit ||\n");

printf("===============================\n");

printf("\n");

printf("Input pilihan: ");

scanf("%d", &mChoice);

switch(mChoice){

case 1: tambahPeserta();

break;

case 2: if (isPesertaEmpty(P) == 0)

lihatPesertaIndiv();

break;

case 3: login();

break;

case 4: lihatSeluruhPeserta();

break;

case 0: break;

default: printf("Input yang Anda masukkan salah");

getch();

break;

}

}

return 0;

}

* **PON .C**

/\* PON.c \*/

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include "PON.H"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LIST KATEGORI \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Melakukan alokasi untuk elemen kategori baru \*/

adrsKategori alokasiKategori(string X) {

adrsKategori P;

P = (adrsKategori)malloc(sizeof(ElmtKategori));

if (P != nil) {

strcpy(kategori(P), X);

next(P) = nil;

firstLomba(P) = nil;

}

return P;

}

/\* Melakukan dealokasi terhadap address dari suatu ElmtKategori \*/

void dealokasiKategori(adrsKategori P) {

free(P);

}

/\* Melakukan alokasi untuk elemen lomba baru \*/

adrsLomba alokasiLomba(string X) {

adrsLomba P;

P = (adrsLomba) malloc(sizeof(ElmtLomba));

if (P != nil) {

strcpy(lomba(P), X);

next(P) = nil;

}

return P;

}

/\* Melakukan dealokasi terhadap address dari suatu ElmtLomba \*/

void dealokasiLomba(adrsLomba P) {

free(P);

}

/\* Inisialisasi pembuatan list kategori \*/

void createListKategori(ListKategori \*K) {

first(\*K) = nil;

}

/\* Menambahakan elemen kategori yang baru ke akhir dari list \*/

void addKategori(ListKategori \*K, string X) {

adrsKategori P, last;

P = alokasiKategori(X);

if (P != nil) {

if (first(\*K) == nil) {

first(\*K) = P;

} else {

last = first(\*K);

while (next(last) != nil) {

last = next(last);

}

next(last) = P;

}

} else

printf("Error: Memori penuh.");

}

/\* Menambahkan elemen lomba yang baru ke akhir dari list lomba \*/

void addLomba(ListKategori \*K, string kategori, string lomba) {

adrsLomba P, last;

adrsKategori C;

P = alokasiLomba(lomba);

if (P != nil) {

if (first(\*K) == nil) {

printf("Error: List Kategori kosong");

} else {

C = first(\*K);

while (C != nil && strcmpi(kategori(C), kategori) != 0) {

C = next(C);

}

if (C != nil && strcmpi(kategori(C), kategori) == 0) {

if (firstLomba(C) == nil) {

firstLomba(C) = P;

} else {

last = firstLomba(C);

while (next(last) != nil) {

last = next(last);

}

next(last) = P;

}

} else

printf("Kategori yang Anda inputkan tidak ada.");

}

}

}

/\* Mengambil address dari suatu kategori \*/

adrsKategori getAddressKategori(ListKategori K, string kategori){

adrsKategori P;

int found;

P = first(K);

while (P != nil && found != 1){

if (strcmpi(kategori(P), kategori) == 0){

found = 1;

} else

P = next(P);

}

if (found == 1)

return P;

else

return nil;

}

/\* Mengambil address dari suatu lomba

\* Mengembalikan nilai nil jika lomba yang dicari tidak ditemukan

\*/

adrsLomba getAddressLomba(ListKategori K, string kategori, string lomba) {

adrsKategori C;

adrsLomba L;

int foundKategori, foundLomba;

foundKategori = 0;

foundLomba = 0;

C = first(K);

while (C != nil && foundKategori != 1) {

if (strcmpi(kategori(C), kategori) == 0){

foundKategori = 1;

L = firstLomba(C);

} else

C = next(C);

}

if (foundKategori == 1) {

while (L != nil && foundLomba != 1){

if (strcmpi(lomba(L), lomba) == 0){

foundLomba = 1;

} else

L = next(L);

}

}

if (foundLomba == 1){

return L;

} else

return nil;

}

/\* Mengambil jumlah lomba yang ada \*/

int getSumLomba(ListKategori K) {

adrsKategori C;

adrsLomba L;

int i = 0;

if (first(K) != nil) {

C = first(K);

while (C != nil) {

if (firstLomba(C) != nil) {

L = firstLomba(C);

while (L != nil) {

i++;

L = next(L);

}

}

C = next(C);

}

}

return i;

}

/\* Mengambil string kategori dari suatu lomba \*/

void getKategoriOfLomba(ListKategori K, string lomba, string \*out){ // BUG

adrsKategori C;

adrsLomba L;

int found = 0;

if (first(K) == nil) {

printf("List kategori masih kosong");

} else {

C = first(K);

while (C != nil && found != 1) {

if (firstLomba(C) != nil) {

L = firstLomba(C);

while (L != nil && found != 1) {

if (strcmpi(lomba(L), lomba) == 0){

strcpy(\*out, kategori(C));

found = 1;

printf("Kategori ditemukan");

} else

L = next(L);

}

}

C = next(C);

}

}

}

/\* Mengambil string lomba dari addressnya \*/

void getStringLomba(ListKategori K, adrsLomba Q, string \*out){

adrsKategori C;

adrsLomba L;

int found;

found = 0;

C = first(K);

while (C != nil && found != 1){

if (firstLomba(C) != nil){

L = firstLomba(C);

while (L != nil && found != 1){

if (L == Q){

found = 1;

strcpy(\*out, lomba(L));

} else

L = next(L);

}

}

C = next(C);

}

}

/\* Memeriksa jika suatu kategori telah ada pada list apa belum

\* Mengembalikan nilai 1 jika ada, dan 0 jika belum ada

\*/

int checkKategoriExistence(ListKategori K, string kategori){

adrsKategori P;

P = getAddressKategori(K, kategori);

if (P == nil){

return 0;

} else

return 1;

}

/\* Memeriksa jika suatu lomba telah ada pada list apa belum \*/

int checkLombaExistence(ListKategori K, string kategori, string lomba){

adrsLomba N;

N = getAddressLomba(K, kategori, lomba);

if (N == nil){

return 0;

} else

return 1;

}

/\* Mencetak seluruh kategori yang ada \*/

void printAllKategori(ListKategori K) {

adrsKategori C;

int i = 1;

if (first(K) == nil) {

printf("List kategori masih kosong");

} else {

C = first(K);

while (C != nil) {

printf("%d. %s \n", i, kategori(C));

C = next(C);

i++;

}

}

}

/\* Mencetak seluruh lomba yang ada dalam kategori tertentu \*/

void printLombaInKategori(ListKategori K, string kategori, int isNumbered) {

adrsKategori C;

adrsLomba L;

int i = 1;

if (first(K) == nil) {

printf("List kategori masih kosong");

} else {

C = first(K);

while (C != nil && strcmpi(kategori(C), kategori) != 0) {

C = next(C);

}

if (C != nil && strcmpi(kategori(C), kategori) == 0) {

if (firstLomba(C) == nil) {

printf("List lomba dari kategori %s masih kosong", kategori);

} else {

L = firstLomba(C);

while (L != nil) {

if(isNumbered == 1){

printf("%d. %s \n", i, lomba(L));

} else if (isNumbered == 0)

printf("- %s \n", lomba(L));

L = next(L);

i++;

}

}

}

}

}

/\* Mencetak seluruh lomba yang ada \*/

void printAllLomba(ListKategori K) {

adrsKategori C;

adrsLomba L;

int i = 1;

if (first(K) == nil) {

printf("List kategori masih kosong");

} else {

C = first(K);

while (C != nil) {

if (firstLomba(C) != nil) {

L = firstLomba(C);

while (L != nil) {

printf("%d. %s \n", i, lomba(L));

i++;

L = next(L);

if (i%10 == 0){

getch();

}

}

}

C = next(C);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LIST DAERAH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Melakukan alokasi untuk elemen daerah baru \*/

adrsDaerah alokasiDaerah(string X) {

adrsDaerah P;

P = (adrsDaerah) malloc(sizeof(ElmtDaerah));

if (P != NULL) {

strcpy(daerah(P), X);

next(P) = nil;

}

return P;

}

/\* Melakukan dealokasi terhadap address dari suatu ElmtDaerah \*/

void dealokasiDaerah(adrsDaerah P) {

free(P);

}

/\* Inisialisasi pembuatan list peserta \*/

void createListDaerah(ListDaerah \*L) {

first(\*L) = nil;

}

/\* Menambahkan elemen daerah yang baru ke akhir dari list \*/

void addDaerah(ListDaerah \*L, string X) {

adrsDaerah P, last;

P = alokasiDaerah(X);

if (P != nil) {

if (first(\*L) == nil) {

first(\*L) = P;

} else {

last = first(\*L);

while (next(last) != nil) {

last = next(last);

}

next(last) = P;

}

} else

printf("Error: Memori penuh.");

}

/\* Mengambil address dari suatu daerah

\* Mengembalikan nilai nil jika daerah yang dicari tidak ditemukan

\*/

adrsDaerah getAddressDaerah(ListDaerah L, string daerah) {

adrsDaerah P;

int found;

P = first(L);

while (P != nil && found != 1){

if (strcmpi(daerah(P), daerah) == 0){

found = 1;

} else

P = next(P);

}

if (found == 1)

return P;

else

return nil;

}

/\* Mengambil nama Daerah dari suatu address \*/

void getStringDaerah(ListDaerah L, adrsDaerah P, string \*out){

int found;

adrsDaerah N;

found = 0;

N = first(L);

while (N != nil && found != 1){

if (N == P){

found = 1;

strcpy(\*out, daerah(N));

}

else

N = next(N);

}

}

/\* Mengambil jumlah daerah yang ada \*/

int getSumDaerah(ListDaerah L) {

adrsDaerah N;

int i = 0;

if (first(L) != nil) {

N = first(L);

while (N != nil) {

i++;

N = next(N);

}

}

return i;

}

/\* Mencetak seluruh daerah yang ada \*/

void printAllDaerah(ListDaerah L) {

adrsDaerah N;

int i = 1;

if (first(L) == nil) {

printf("List daerah masih kosong.");

} else {

N = first(L);

while (N != nil) {

printf("%d. %s \n", i, daerah(N));

N = next(N);

i++;

}

}

}

/\* Memeriksa jika suatu daerah telah ada pada list apa belum

\* Mengembalikan nilai 1 jika ada, dan 0 jika belum ada

\*/

int checkDaerahExistence(ListDaerah L, string daerah){

adrsDaerah N;

N = getAddressDaerah(L, daerah);

if (N == nil){

return 0;

} else

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LIST PESERTA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Melakukan alokasi untuk elemen peserta yang baru \*/

adrsPeserta alokasiPeserta(string X, adrsLomba lomba, adrsDaerah daerah) {

adrsPeserta P;

P = (adrsPeserta) malloc(sizeof(ElmtPeserta));

if (P != nil) {

strcpy(peserta(P), X);

next(P) = nil;

daerah(P) = daerah;

lomba(P) = lomba;

}

return P;

}

/\* Melakukan dealokasi terhadap address dari suatu ElmtPeserta \*/

void dealokasiPeserta(adrsPeserta P) {

free(P);

}

/\* Memeriksa jika list peserta kosong \*/

int isPesertaEmpty(ListPeserta L){

if (first(L) == nil)

return 1;

else

return 0;

}

/\* Inisialisasi pembuatan list peserta \*/

void createListPeserta(ListPeserta \*L) {

first(\*L) = nil;

}

/\* Menambahkan elemen peserta yang baru ke akhir dari list \*/

void addPeserta(ListPeserta \*L, string X, adrsLomba lomba, adrsDaerah daerah) {

adrsPeserta P, last;

P = alokasiPeserta(X, lomba, daerah);

if (P != nil) {

if (first(\*L) == nil) {

first(\*L) = P;

} else {

last = first(\*L);

while (next(last) != nil) {

last = next(last);

}

next(last) = P;

}

} else

printf("Error: Memori penuh.");

}

/\* Menghapus peserta yang telah selesai mengikuti lomba \*/

void deletePeserta(ListPeserta \*L, string peserta, string \*out) {

adrsPeserta P, X;

if (first(\*L) == nil) {

printf("List peserta masih kosong.\n");

} else if (strcmpi(peserta(first(\*L)), peserta) == 0){

X = first(\*L);

strcpy( \*out, peserta(first(\*L)));

first(\*L) = next(first(\*L));

dealokasiPeserta(X);

} else {

P = first(\*L);

while (P != nil && strcmpi(peserta(next(P)), peserta) != 0) {

P = next(P);

}

if (P != nil) {

X = next(P);

strcpy(\*out, peserta(next(P)));

next(P) = next(next(P));

dealokasiPeserta(X);

}

}

}

/\* Menghapus daerah yang telah selesai mengikuti lomba \*/

void deleteDaerah(ListDaerah \*N, string daerah, string \*out) {

adrsDaerah P, X;

if (first(\*N) == nil) {

printf("List daerah masih kosong.\n");

} else if (strcmpi(daerah(first(\*N)), daerah) == 0){

X = first(\*N);

strcpy( \*out, daerah(first(\*N)));

first(\*N) = next(first(\*N));

dealokasiDaerah(X);

} else {

P = first(\*N);

while (P != nil && strcmpi(daerah(next(P)), daerah) != 0) {

P = next(P);

}

if (P != nil) {

X = next(P);

strcpy(\*out, daerah(next(P)));

next(P) = next(next(P));

dealokasiDaerah(X);

}

}

}

/\* Mengambil jumlah peserta yang ada \*/

int getSumPeserta(ListPeserta L) {

adrsPeserta P;

int i = 0;

if (first(L) != nil) {

P = first(L);

while (P != nil) {

i++;

P = next(P);

}

}

return i;

}

/\* Mengambil address dari peserta \*/

adrsPeserta getAddressPeserta(ListPeserta L, string peserta){

adrsPeserta P;

int found;

P = first(L);

while (P != nil && found != 1){

if (strcmpi(peserta(P), peserta) == 0){

found = 1;

} else

P = next(P);

}

if (found == 1)

return P;

else

return nil;

}

/\* Memeriksa jika seorang peserta telah ada pada list apa belum

\* Mengembalikan nilai 1 jika ada, dan 0 jika belum ada

\*/

int checkPesertaExistence(ListPeserta L, string peserta){

adrsPeserta P;

P = getAddressPeserta(L, peserta);

if (P == nil){

return 0;

} else

return 1;

}

/\* Mencetak seluruh peserta yang ada \*/

void printAllPeserta(ListPeserta L, ListDaerah N, ListKategori K, int isDetail) {

adrsPeserta P;

string daerah, lomba;

int i = 1;

if (first(L) == nil) {

printf("List peserta masih kosong.\n");

} else {

P = first(L);

while (P != nil) {

if (isDetail == 1){

getStringDaerah(N, daerah(P), &daerah);

getStringLomba( K, lomba(P), &lomba); // NEED TO BE FIXED

printf("(%d)\n",i);

printf("Nama Peserta : %s \n", peserta(P));

printf("Lomba : %s \n", lomba);

printf("Daerah : %s \n", daerah);

printf("\n");

} else if (isDetail == 0){

printf("%d. %s\n", i, peserta(P));

}

P = next(P);

i++;

}

}

}

/\* Mencetak detil data seorang peserta berdasarkan namanya \*/

void printPesertaIndiv(ListKategori K, ListDaerah N, ListPeserta L, string peserta){

adrsPeserta P;

string lomba, daerah;

P = getAddressPeserta( L, peserta);

getStringLomba( K, lomba(P), &lomba);

getStringDaerah(N, daerah(P), &daerah);

printf("Nama Peserta : %s\n", peserta(P));

printf("Lomba : %s\n", lomba);

printf("Daerah : %s\n", daerah);

}

/\* Mencetak peserta yang mengikuti lomba (adrsLomba) tertentu \*/

void printPesertaInLomba(ListKategori K, ListPeserta L, adrsLomba lomba) {

adrsPeserta P;

int i = 1;

if (first(L) == nil) {

printf("List peserta masih kosong.");

} else {

P = first(L);

while (P != nil) {

if (lomba(P) == lomba) {

printf("%d. %s \n", i, peserta(P));

i++;

}

P = next(P);

}

}

}

/\* Mencetak jumlah peserta yang mengikuti lomba (adrsLomba) tertentu \*/

int printSumPesertaInLomba(ListPeserta L, adrsLomba lomba) {

adrsPeserta P;

int i = 0;

if (first(L) != nil) {

P = first(L);

while (P != nil) {

if (lomba(P) == lomba) {

i++;

}

P = next(P);

}

}

return i;

}

/\* Mencetak peserta yang berasal dari daerah (adrsDaerah) tertentu \*/

void printPesertaInDaerah(ListPeserta L, adrsDaerah daerah) {

adrsPeserta P;

int i = 1;

if (first(L) == nil) {

printf("List peserta masih kosong.");

} else {

P = first(L);

while (P != nil) {

if (daerah(P) == daerah) {

printf("%d. %s \n", i, peserta(P));

i++;

}

P = next(P);

}

}

}

/\* Mencetak jumlah peserta yang berasal dari daerah (adrsDaerah) tertentu \*/

int printSumPesertaInDaerah(ListPeserta L, adrsDaerah daerah) {

adrsPeserta P;

int i = 0;

if (first(L) != nil) {

P = first(L);

while (P != nil) {

if (daerah(P) == daerah) {

i++;

}

P = next(P);

}

}

return i;

}