LAPORAN PRAKTIKUM

POSTTEST 6

ALGORITMA PEMROGRAMAN LANJUT

Disusun oleh:

Muhammad Rafli Pernanda 2409106040

Kelas A2 ‘24

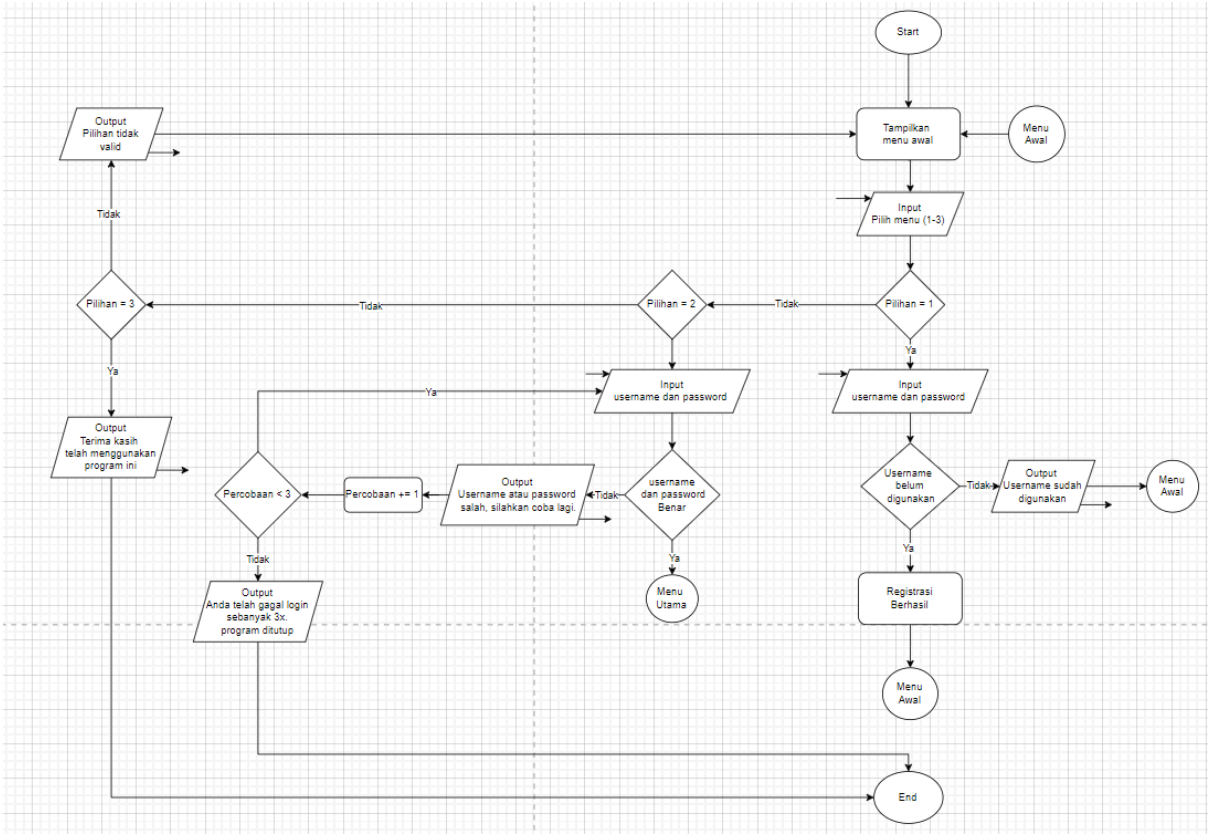
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

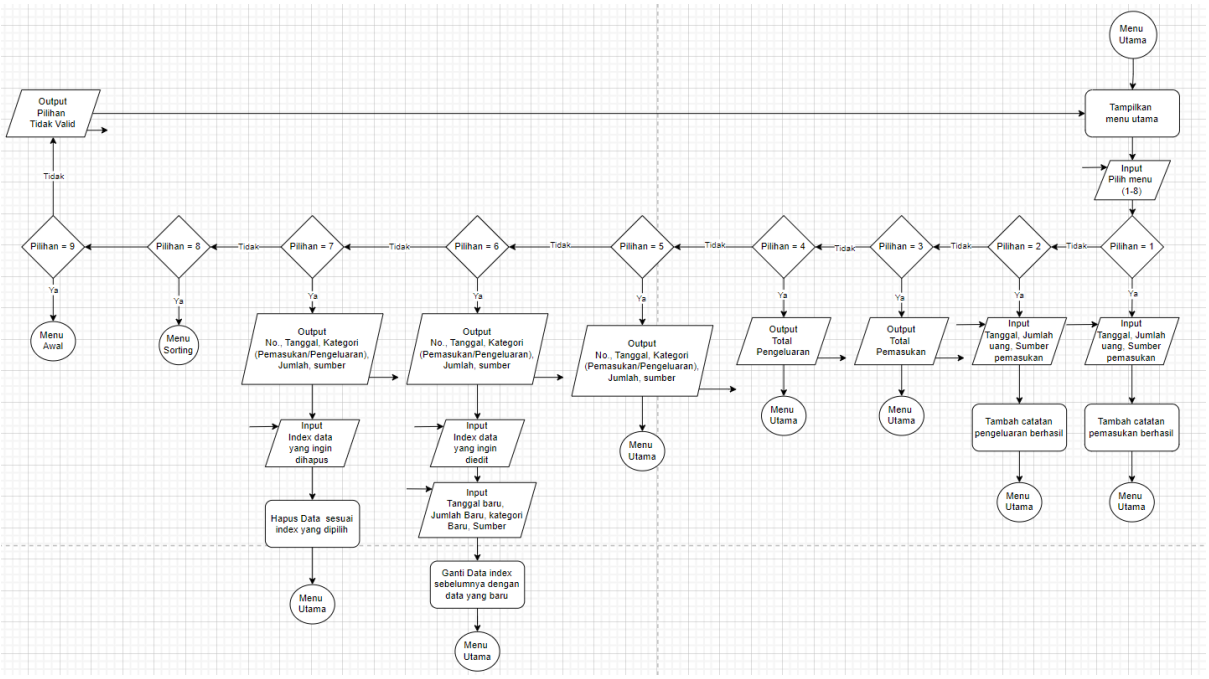
SAMARINDA

2025

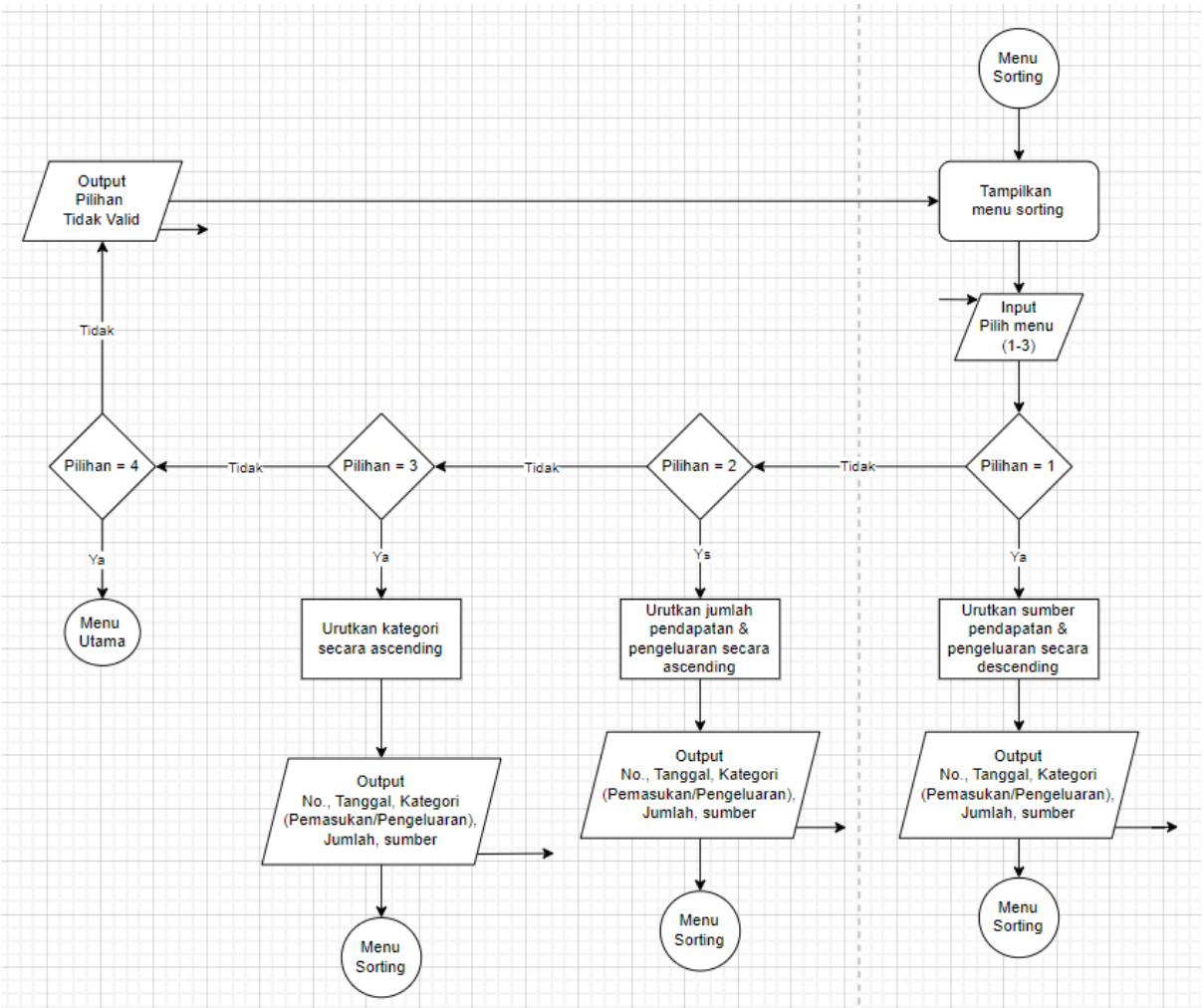
1. Flowchart



Gambar 1.1 Flowchart Menu Awal



Gambar 1.2 Flowchart Menu Utama



Gambar 1.3 Flowchart Menu Sorting

2. Analisis Program

Program ini dirancang untuk membantu pengguna dalam mencatat dan mengelola transaksi keuangan pribadi. Pengguna dapat melakukan registrasi akun terlebih dahulu, kemudian login untuk mengakses fitur-fitur utama. Sistem keamanan juga diterapkan dengan membatasi percobaan login sebanyak tiga kali untuk menghindari penyalahgunaan.

Setelah berhasil login, pengguna dapat menambahkan catatan pemasukan maupun pengeluaran yang terdiri dari informasi tanggal, jumlah uang, sumber dana, dan kategori. Selain itu, pengguna juga dapat melihat total pemasukan maupun pengeluaran, menampilkan semua catatan, serta melakukan pengeditan atau penghapusan data jika diperlukan. Untuk mempermudah proses pemilihan saat edit atau hapus, program akan menampilkan seluruh catatan terlebih dahulu.

Program menggunakan pendekatan modular dengan memanfaatkan fungsi-fungsi terpisah yang menerima parameter pointer (`Catatan*` dan `int*`) agar data lebih fleksibel diproses tanpa harus bergantung pada variabel global.

Fitur tambahan yang menjadi keunggulan dari program ini adalah adanya menu khusus untuk sorting data. Pengguna dapat mengurutkan catatan berdasarkan beberapa kriteria:

- Sumber (menggunakan algoritma Selection Sort, urutan menurun/descending),
- Jumlah uang (menggunakan algoritma Bubble Sort, urutan naik/ascending), dan
- Kategori (menggunakan algoritma Merge Sort, urutan naik/ascending).

Dengan penerapan algoritma-algoritma tersebut, pengguna dapat mengelola dan menganalisis data keuangan mereka dengan lebih baik dan terstruktur. Program ini diharapkan dapat menjadi alat bantu pencatatan keuangan yang sederhana namun fungsional.

3. Source Code

A. Deklarasi Variable

Bagian ini berisi deklarasi struct, array global, dan konstanta yang digunakan untuk menyimpan data pengguna serta catatan keuangan.

Source Code:

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

const int MAX_USERS = 100;
const int MAX_CATATAN = 100;

struct User {
    string username;
    string password;
};

struct Catatan {
    string tanggal;
    string kategori;
    double jumlah;
    string sumber;
};

User users[MAX_USERS] = {{"pernanda", "040"}};
Catatan catatan[MAX_CATATAN];

int totalUsers = 1;
int totalCatatan = 0;
```

B. Menu Awal

Menampilkan tampilan awal program dan meminta input dari user untuk memilih registrasi, login, atau keluar dari program.

Source Code:

```
void MenuAwal() {
    cout << "==================================\n";
    cout << "|           MENU AWAL            |\n";
    cout << "==================================\n";
    cout << "| 1. Registrasi                  |\n";
```

```
    cout << "| 2. Login                    |\n";
    cout << "| 3. Keluar                   |\n";
    cout << "===============================\n";
}
```

C. Registrasi User

Digunakan untuk menambahkan akun user baru ke dalam array *users* jika username belum pernah digunakan

Source Code:

```
void registrasi() {
    if (totalUsers >= MAX_USERS) {
        cout << "User sudah penuh!\n";
        return;
    }

    string uname, pass;
    cout << "Masukkan username baru: ";
    cin >> uname;

    if (usernameSudahAda(uname)) {
        cout << "Username sudah digunakan!\n";
        return;
    }

    cout << "Masukkan password: ";
    cin >> pass;

    users[totalUsers++] = {uname, pass};
    cout << "Registrasi berhasil!\n\n";
}
```

D. Login User

Memverifikasi username dan password, jika username dan password benar, maka akan masuk ke dalam menu utama, serta membatasi percobaan login maksimal tiga kali.

Source Code:

```
bool login(string& username) {
    string inputUser, inputPass;
    int attempts = 0;

    while (attempts < 3) {
```

```
        cout << "Masukkan Username: ";
        cin >> inputUser;
        cout << "Masukkan Password: ";
        cin >> inputPass;

        for (int i = 0; i < totalUsers; i++) {
            if (users[i].username == inputUser && users[i].password ==
inputPass) {
                username = inputUser;
                cout << "Login berhasil! Selamat datang, " << inputUser <<
"!\n\n";
                return true;
            }
        }

        attempts++;
        cout << "Login gagal! Sisa percobaan: " << (3 - attempts) << "\n\n";
    }

    return false;
}
```

E. Menambah Catatan

Digunakan untuk mencatat transaksi baru, baik pemasukan maupun pengeluaran .

Source Code:

```
void tambahCatatan(bool pemasukan, Catatan* catatan, int* totalCatatan) {
    if (*totalCatatan >= MAX_CATATAN) {
        cout << "Penyimpanan penuh!\n";
        return;
    }

    cin.ignore();
    cout << "Masukkan tanggal (YYYY-MM-DD): ";
    getline(cin, catatan[*totalCatatan].tanggal);
    cout << "Masukkan jumlah uang: ";
    cin >> catatan[*totalCatatan].jumlah;
    cin.ignore();
    cout << "Masukkan sumber: ";
    getline(cin, catatan[*totalCatatan].sumber);
    catatan[*totalCatatan].kategori = pemasukan ? "Pemasukan" : "Pengeluaran";

    (*totalCatatan)++;
    cout << "Catatan berhasil ditambahkan!\n\n";
}
```

F. Melihat Total Pemasukan / Pengeluaran

Menjumlahkan semua nilai jumlah dari catatan berdasarkan kategori tertentu.

Source Code:

```
void lihatTotal(bool pemasukan, Catatan* catatan, int* totalCatatan) {
    double total = 0;
    string tipe = pemasukan ? "Pemasukan" : "Pengeluaran";

    for (int i = 0; i < *totalCatatan; i++) {
        if (catatan[i].kategori == tipe) {
            total += catatan[i].jumlah;
        }
    }

    cout << "Total " << tipe << ": Rp" << total << "\n\n";
}
```

G. Menampilkan Semua Catatan

Menampilkan seluruh catatan yang telah ditambahkan dalam bentuk tabel.

Source Code:

```
void tampilkanSemuaCatatan(Catatan* catatan, int* totalCatatan) {
    if (*totalCatatan == 0) {
        cout << "Belum ada catatan.\n\n";
        return;
    }

    cout << "\n================== Daftar Catatan ==================\n";
    cout <<
"+----+------------+------------+------------+-----------------+\n";
    cout << "| No |   Tanggal  |  Kategori  |   Jumlah   |      Sumber
|\n";
    cout <<
"+----+------------+------------+------------+-----------------+\n";

    for (int i = 0; i < *totalCatatan; i++) {
        cout << "| " << setw(2) << i + 1
             << " | " << setw(10) << catatan[i].tanggal
             << " | " << setw(10) << catatan[i].kategori
             << " | " << setw(10) << catatan[i].jumlah
             << " | " << setw(16) << catatan[i].sumber
             << " |\n";
    }
```

```
    cout <<
"+----+------------+-----------+-----------+-----------------+\n\n";
}
```

H. Memperbarui Catatan

Menampilkan semua catatan terlebih dahulu, lalu memperbarui data sesuai indeks yang dipilih user.

Source Code:

```
void editCatatan(Catatan* catatan, int* totalCatatan) {
    tampilkanSemuaCatatan(catatan, totalCatatan);

    if (*totalCatatan == 0) return;

    int index;
    cout << "Masukkan nomor catatan yang ingin diedit: ";
    cin >> index;

    if (index < 1 || index > *totalCatatan) {
        cout << "Nomor tidak valid!\n\n";
        return;
    }

    index--;
    cin.ignore();
    cout << "Masukkan tanggal baru: ";
    getline(cin, catatan[index].tanggal);
    cout << "Masukkan jumlah baru: ";
    cin >> catatan[index].jumlah;
    cin.ignore();
    cout << "Masukkan sumber baru: ";
    getline(cin, catatan[index].sumber);

    cout << "Catatan berhasil diperbarui!\n\n";
}
```

I. Menghapus Catatan

Menampilkan semua catatan terlebih dahulu, lalu menghapus catatan sesuai indeks yang dimasukkan user.

Source Code:

```
void hapusCatatan(Catatan* catatan, int* totalCatatan) {
```

```
    tampilkanSemuaCatatan(catatan, totalCatatan);

    if (*totalCatatan == 0) return;

    int index;
    cout << "Masukkan nomor catatan yang ingin dihapus: ";
    cin >> index;

    if (index < 1 || index > *totalCatatan) {
        cout << "Nomor tidak valid!\n\n";
        return;
    }

    for (int i = index - 1; i < *totalCatatan - 1; i++) {
        catatan[i] = catatan[i + 1];
    }
    (*totalCatatan)--;

    cout << "Catatan berhasil dihapus!\n\n";
}
```

J. Sorting Catatan

Fitur ini digunakan untuk mengurutkan data catatan keuangan berdasarkan kriteria tertentu agar data lebih terstruktur dan mudah dianalisis. Program menyediakan tiga metode pengurutan dengan algoritma berbeda:

- Selection Sort untuk mengurutkan catatan berdasarkan sumber secara menurun (descending).
- Bubble Sort untuk mengurutkan catatan berdasarkan jumlah uang secara menaik (ascending).
- Merge Sort untuk mengurutkan catatan berdasarkan kategori secara menaik (ascending).

Source Code:

```
// Fungsi untuk menukar dua catatan
void swapCatatan(Catatan &a, Catatan &b) {
    Catatan temp = a;
    a = b;
    b = temp;
}

// 1. Selection Sort untuk mengurutkan sumber secara descending
```

```
void sortBySumberDescending(Catatan* catatan, int totalCatatan) {
    for (int currentIndex = 0; currentIndex < totalCatatan - 1; currentIndex++)
{
        int indexSumberTerbesar = currentIndex;
        for (int compareIndex = currentIndex + 1; compareIndex < totalCatatan;
compareIndex++) {
            if (catatan[compareIndex].sumber >
catatan[indexSumberTerbesar].sumber) {
                indexSumberTerbesar = compareIndex;
            }
        }
        if (indexSumberTerbesar != currentIndex) {
            swapCatatan(catatan[currentIndex], catatan[indexSumberTerbesar]);
        }
    }
}

// 2. Bubble Sort untuk mengurutkan jumlah secara ascending
void sortByJumlahAscending(Catatan* catatan, int totalCatatan) {
    for (int passIndex = 0; passIndex < totalCatatan - 1; passIndex++) {
        bool isAdaPertukaran = false;
        for (int compareIndex = 0; compareIndex < totalCatatan - passIndex - 1;
compareIndex++) {
            if (catatan[compareIndex].jumlah > catatan[compareIndex + 1].jumlah)
{
                swapCatatan(catatan[compareIndex], catatan[compareIndex + 1]);
                isAdaPertukaran = true;
            }
        }
        if (!isAdaPertukaran) break;
    }
}

// 3. Merge Sort untuk mengurutkan kategori secara ascending
void merge(Catatan* catatan, int indexAwal, int indexTengah, int indexAkhir) {
    int lenghtKanan = indexTengah - indexAwal + 1;
    int lenghtKiri = indexAkhir - indexTengah;

    Catatan* arrayKiri = new Catatan[lenghtKanan];
    Catatan* arrayKanan = new Catatan[lenghtKiri];

    for (int isiArrayKiri = 0; isiArrayKiri < lenghtKanan; isiArrayKiri++)
        arrayKiri[isiArrayKiri] = catatan[indexAwal + isiArrayKiri];
    for (int isiArrayKanan = 0; isiArrayKanan < lenghtKiri; isiArrayKanan++)
        arrayKanan[isiArrayKanan] = catatan[indexTengah + 1 + isiArrayKanan];

    int indexGabung = indexAwal, indexLeft = 0, indexRight = 0;
    while (indexLeft < lenghtKanan && indexRight < lenghtKiri) {
        if (arrayKiri[indexLeft].kategori <= arrayKanan[indexRight].kategori) {
            catatan[indexGabung] = arrayKiri[indexLeft];
            indexLeft++;
```

```
        } else {
            catatan[indexGabung] = arrayKanan[indexRight];
            indexRight++;
        }
        indexGabung++;
    }

    while (indexLeft < lenghtKanan) {
        catatan[indexGabung] = arrayKiri[indexLeft];
        indexLeft++;
        indexGabung++;
    }
    while (indexRight < lenghtKiri) {
        catatan[indexGabung] = arrayKanan[indexRight];
        indexRight++;
        indexGabung++;
    }

    delete[] arrayKiri;
    delete[] arrayKanan;
}

void mergeSort(Catatan* catatan, int indexAwal, int indexAkhir) {
    if (indexAwal < indexAkhir) {
        int indexTengah = indexAwal + (indexAkhir - indexAwal) / 2;
        mergeSort(catatan, indexAwal, indexTengah);
        mergeSort(catatan, indexTengah + 1, indexAkhir);
        merge(catatan, indexAwal, indexTengah, indexAkhir);
    }
}

void sortByKategoriAscending(Catatan* catatan, int totalCatatan) {
    mergeSort(catatan, 0, totalCatatan - 1);
}
```

K. Menu Sorting

Menu ini merupakan antarmuka yang disediakan agar pengguna dapat memilih jenis pengurutan yang diinginkan. Setelah pengguna memilih salah satu opsi, data akan langsung diproses menggunakan algoritma yang sesuai dan hasilnya ditampilkan ke layar. Menu ini juga memberikan opsi untuk kembali ke menu utama setelah proses pengurutan selesai.

Source Code:

```
void menuSorting(Catatan* catatan, int* totalCatatan) {
    int pilihan;
    while (true) {
```

```
        cout << "========================================\n";
        cout << "|          MENU SORTING CATATAN         |\n";
        cout << "========================================\n";
        cout << "| 1. Urutkan berdasarkan Sumber (Desc)  |\n";
        cout << "| 2. Urutkan berdasarkan Jumlah (Asc)   |\n";
        cout << "| 3. Urutkan berdasarkan Kategori (Asc) |\n";
        cout << "| 4. Kembali ke Menu Utama              |\n";
        cout << "========================================\n";
        cout << "Pilih menu (1-4): ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
            if (*totalCatatan == 0) {
                cout << "Belum ada catatan.\n\n";
                return;
            } else {
                    sortBySumberDescending(catatan, *totalCatatan);
                    cout << "\nCatatan berhasil diurutkan berdasarkan Sumber
(Descending)\n\n";
                    tampilkanSemuaCatatan(catatan, totalCatatan);}
                break;
            case 2:
            if (*totalCatatan == 0) {
                cout << "Belum ada catatan.\n\n";
                return;
            } else {
                    sortByJumlahAscending(catatan, *totalCatatan);
                    cout << "\nCatatan berhasil diurutkan berdasarkan Jumlah
(Ascending)\n\n";
                    tampilkanSemuaCatatan(catatan, totalCatatan);}
                    break;
            case 3:
                if (*totalCatatan == 0) {
                    cout << "Belum ada catatan.\n\n";
                    return;
                } else {
                    sortByKategoriAscending(catatan, *totalCatatan);
                    cout << "\nCatatan berhasil diurutkan berdasarkan Kategori
(Ascending)\n\n";
                    tampilkanSemuaCatatan(catatan, totalCatatan);}
                    break;
            case 4:
                return;
            default:
                cout << "Pilihan tidak valid!\n\n";
        }
    }
}
```

L. Tampilan Menu Utama

Menampilkan menu utama setelah user berhasil login. Berisi berbagai pilihan fitur pengelolaan catatan keuangan.

Source Code:

```
void menuUtama(Catatan* catatan, int* totalCatatan) {
    int pilihan;
    while (true) {
        cout << "========================================\n";
        cout << "|     Aplikasi Catatan Keuangan        |\n";
        cout << "========================================\n";
        cout << "| 1. Tambah Catatan Pemasukan          |\n";
        cout << "| 2. Tambah Catatan Pengeluaran        |\n";
        cout << "| 3. Lihat Total Pemasukan             |\n";
        cout << "| 4. Lihat Total Pengeluaran           |\n";
        cout << "| 5. Lihat Semua Catatan               |\n";
        cout << "| 6. Perbarui/Edit Catatan             |\n";
        cout << "| 7. Hapus Catatan                     |\n";
        cout << "| 8. Logout                            |\n";
        cout << "========================================\n";
        cout << "Pilih menu (1-8): ";
        cin >> pilihan;

        switch (pilihan) {
            case 1: tambahCatatan(true, catatan, totalCatatan); break;
            case 2: tambahCatatan(false, catatan, totalCatatan); break;
            case 3: lihatTotal(true, catatan, totalCatatan); break;
            case 4: lihatTotal(false, catatan, totalCatatan); break;
            case 5: tampilkanSemuaCatatan(catatan, totalCatatan); break;
            case 6: editCatatan(catatan, totalCatatan); break;
            case 7: hapusCatatan(catatan, totalCatatan); break;
            case 8: cout << "\nLogout berhasil!\n\n"; return;
            default: cout << "Pilihan tidak valid!\n\n";
        }
    }
}
```

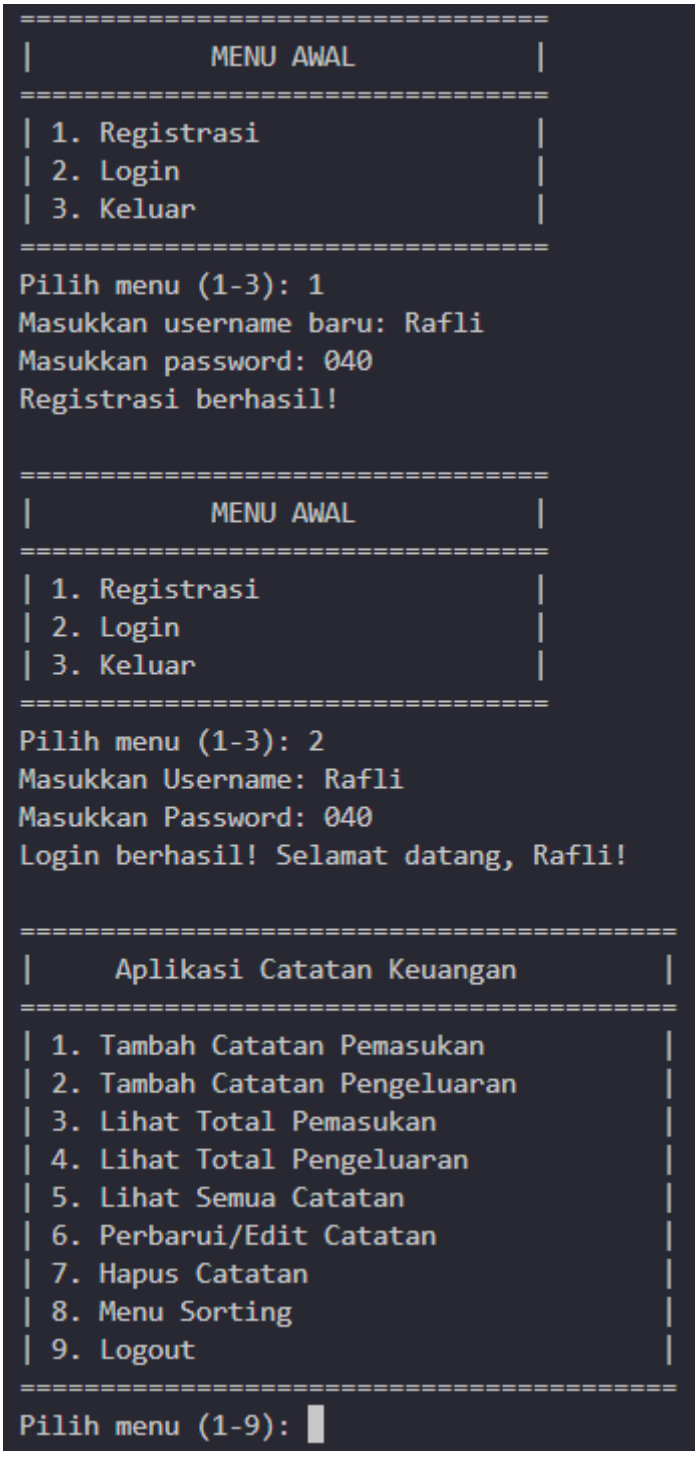
M. Fungsi main()

Fungsi main() merupakan titik awal program berjalan. DI dalamnya terdapat loop utama yang akan terus menampilkan menu awal hingga pengguna memilih untuk keluar. Jika pengguna berhasil login, maka akan diarahkan ke menuUtama() untuk mengakses fitur-fitur utama.

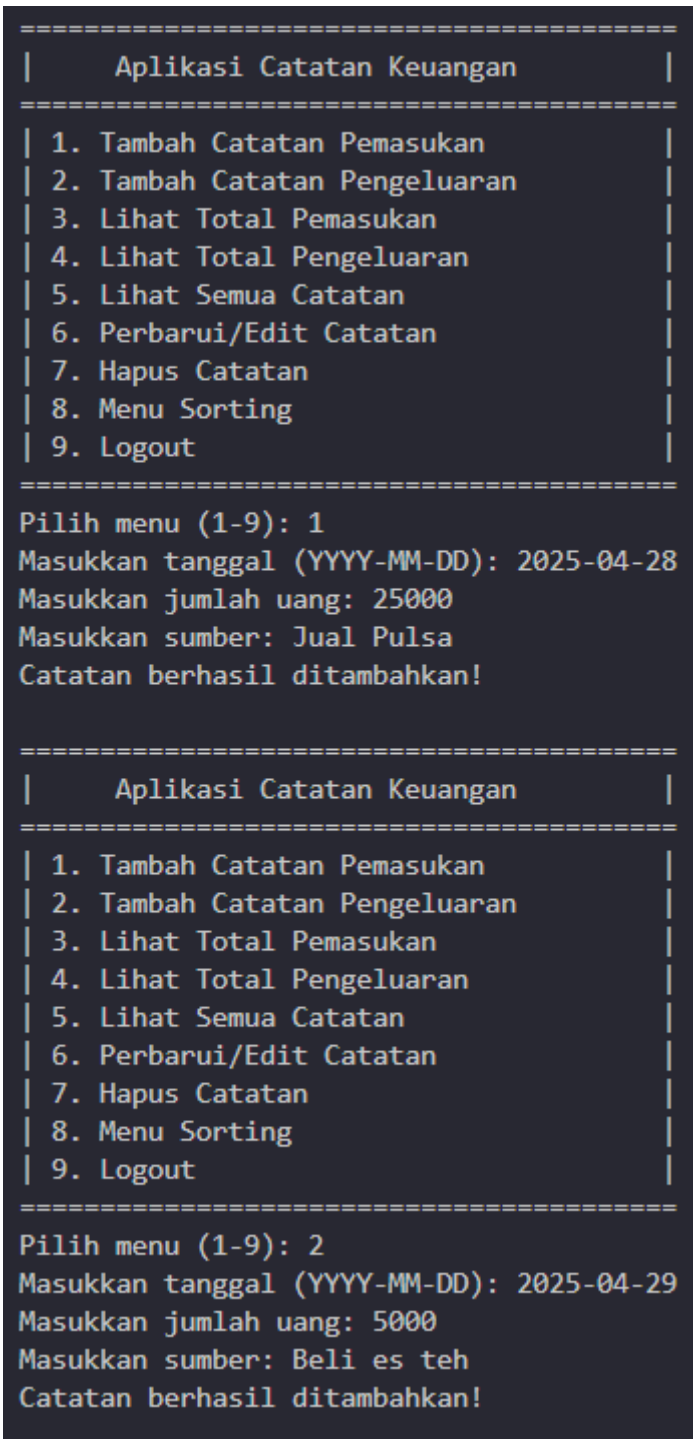
Source Code:

```
int main() {
    int pilihan;
    string usernameLogin;

    while (true) {
        MenuAwal();
        cout << "Pilih menu (1-3): ";
        cin >> pilihan;

        switch (pilihan) {
            case 1: registrasi(); break;
            case 2:
                if (login(usernameLogin)) {
                    menuUtama(catatan, &totalCatatan);
                } else {
                    cout << "Anda gagal login 3 kali. Program dihentikan.\n";
                    return 0;
                }
                break;
            case 3:
                cout << "Terima kasih karena telah menggunakan program ini\n";
                cout << "------------Muhammad Rafli Pernanda------------\n";
                cout << "--------------------2409106040------------------\n";
                return 0;
            default: cout << "Pilihan tidak valid!\n\n";
        }
    }
}
```

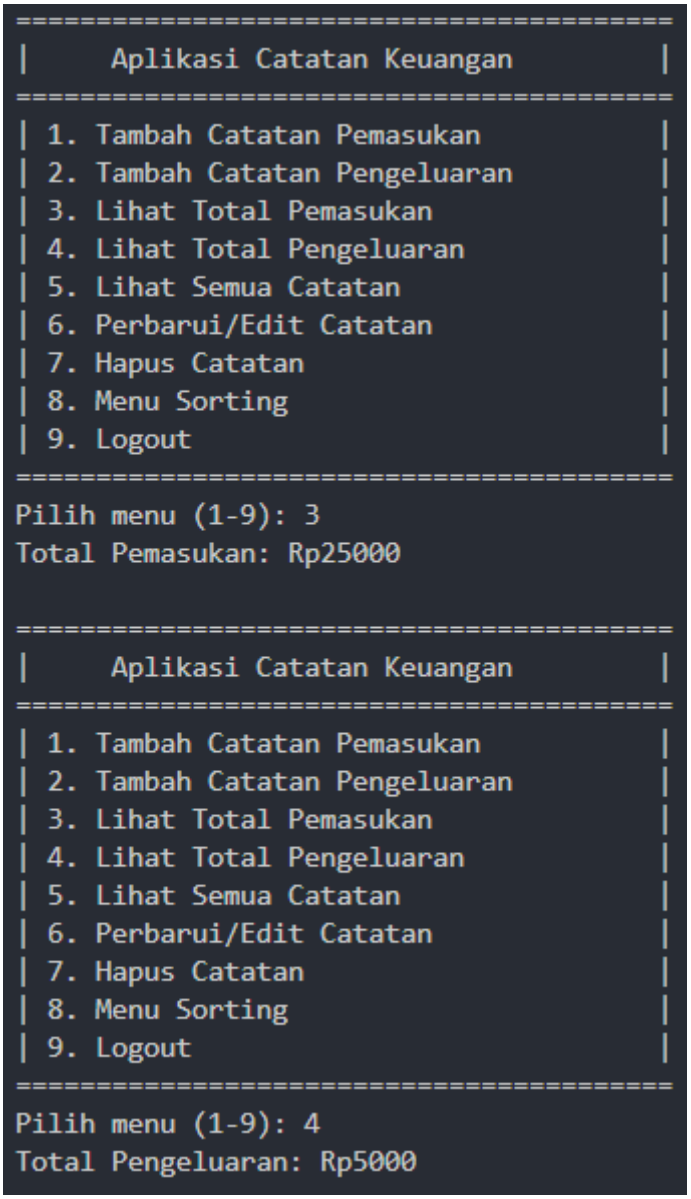
4. Uji Coba dan Hasil Output



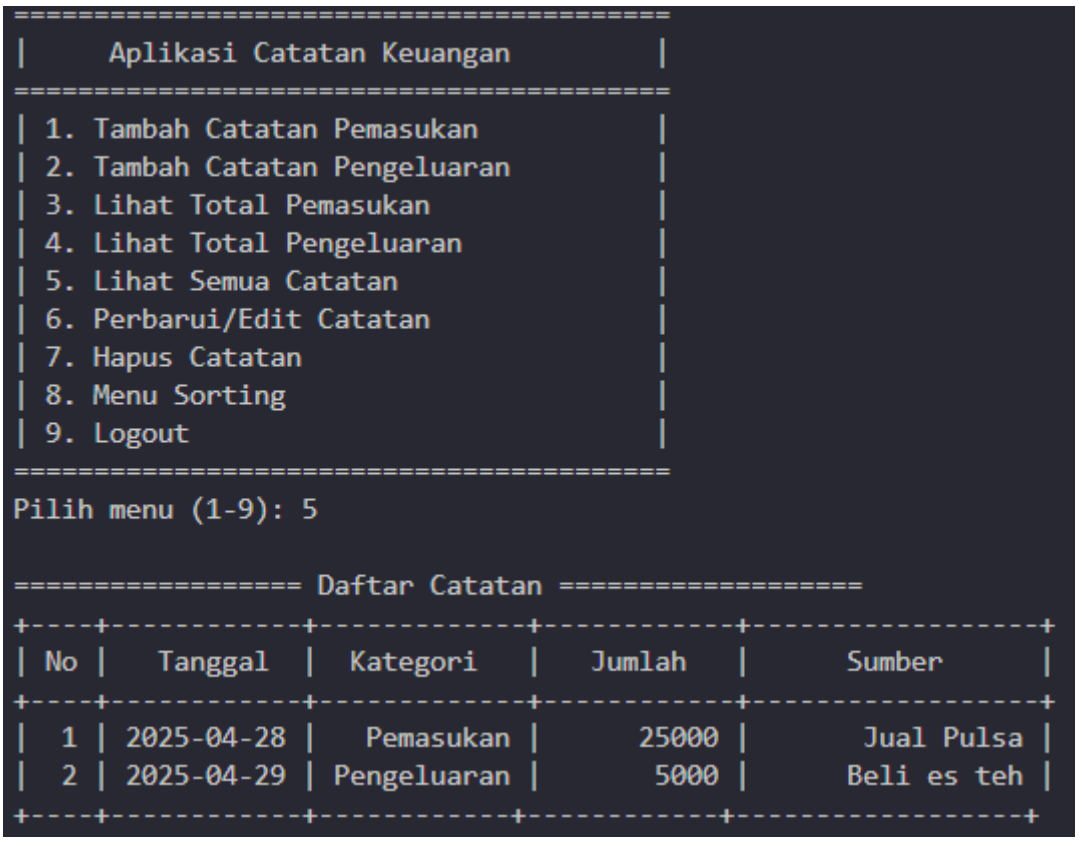
Gambar 4.1 Menu awal, registrasi, dan login



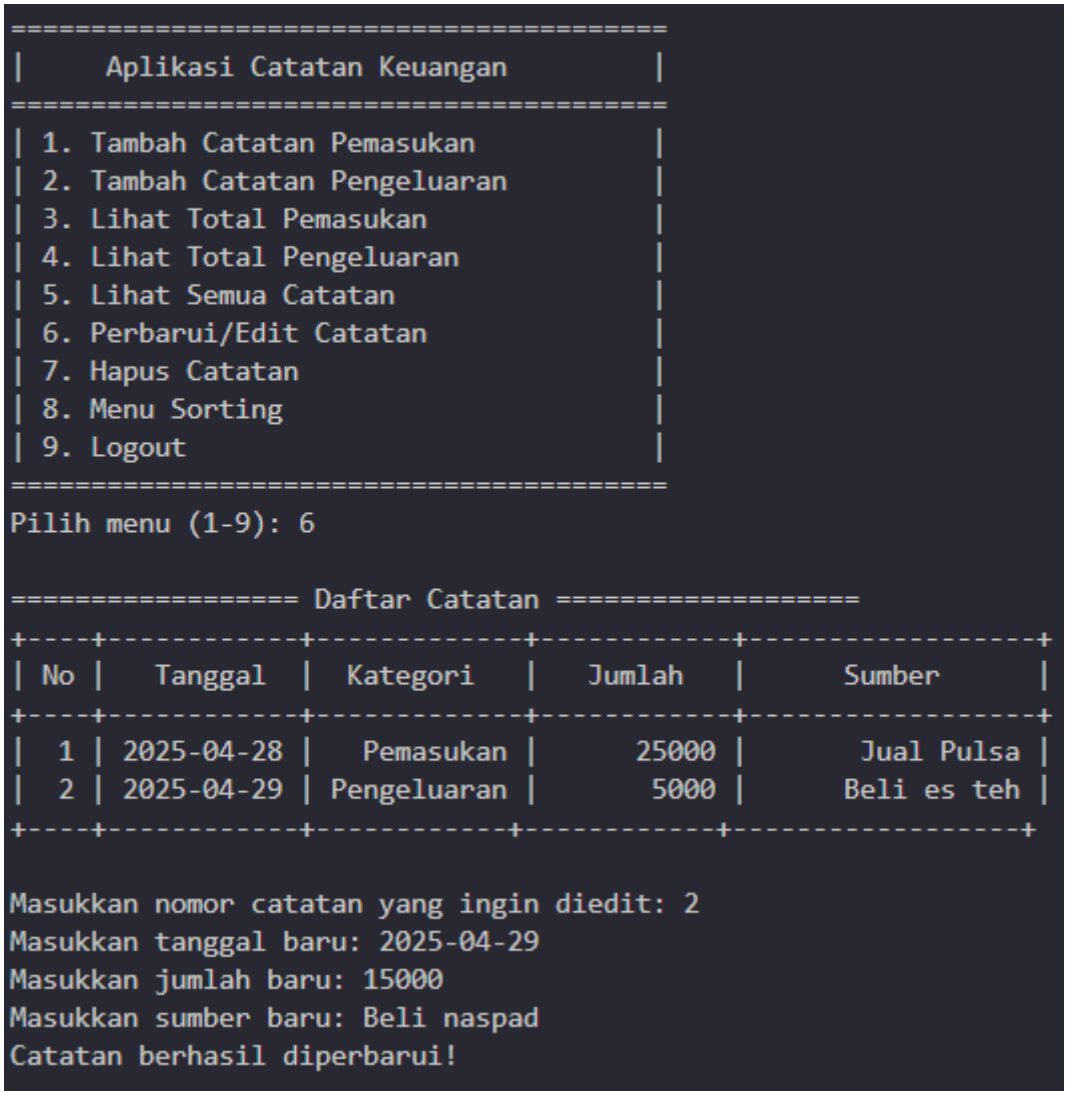
Gambar 4.2 Menambah Catatan Pemasukan & Pengeluaran



Gambar 4.3 Melihat total pemasukan & pengeluaran

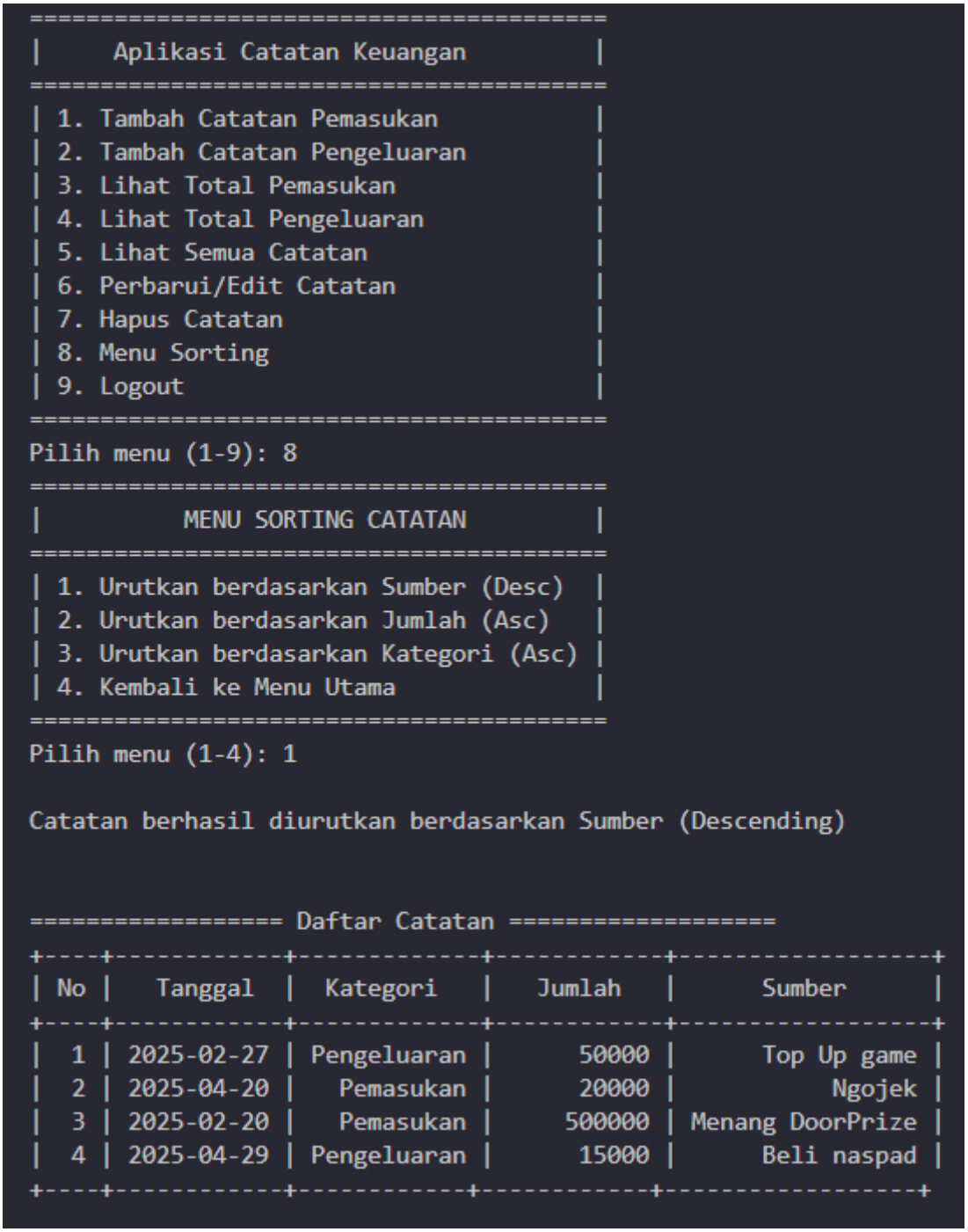


Gambar 4.4 Menampilkan semua catatan



Gambar 4.5 Mengedit catatan

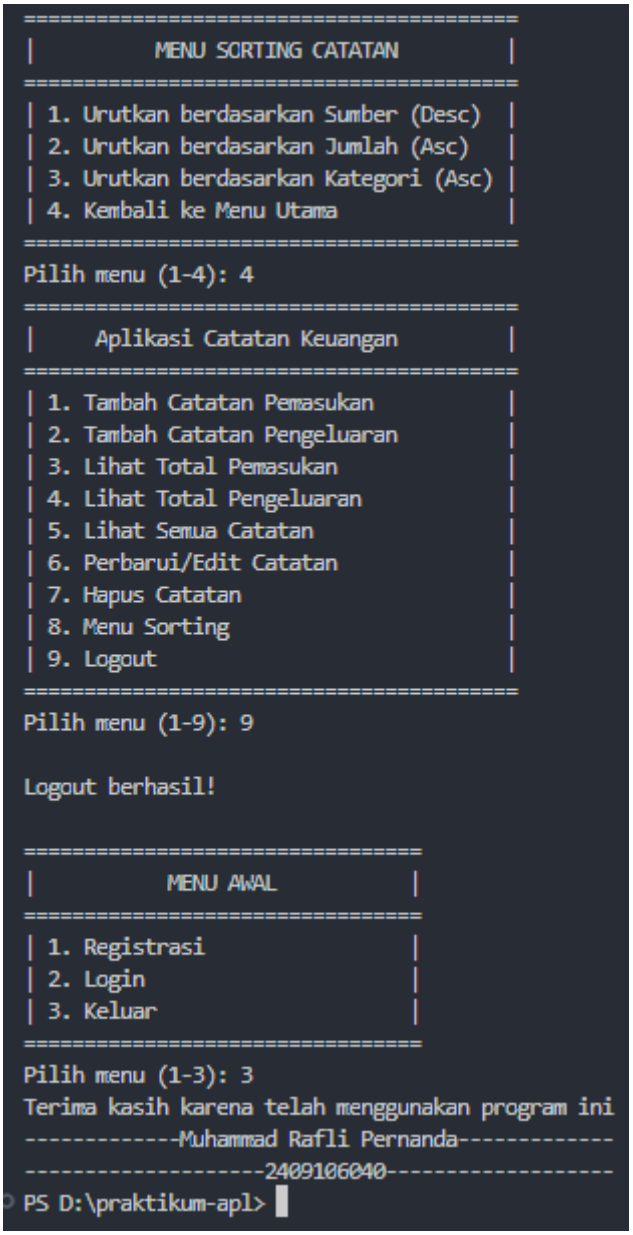
Gambar 4.6 Menghapus catatan



Gambar 4.7 Menu sorting dan sorting sumber pendapatan & pengeluaran secara descending

```
========================================
|          MENU SORTING CATATAN        |
========================================
| 1. Urutkan berdasarkan Sumber (Desc) |
| 2. Urutkan berdasarkan Jumlah (Asc)  |
| 3. Urutkan berdasarkan Kategori (Asc)|
| 4. Kembali ke Menu Utama             |
========================================
Pilih menu (1-4): 2

Catatan berhasil diurutkan berdasarkan Jumlah (Ascending)


================== Daftar Catatan ==================
+----+------------+-------------+-----------+-----------------+
| No |   Tanggal  |  Kategori   |   Jumlah  |      Sumber     |
+----+------------+-------------+-----------+-----------------+
|  1 | 2025-04-29 | Pengeluaran |     15000 |      Beli naspad |
|  2 | 2025-04-20 |   Pemasukan |     20000 |          Ngojek |
|  3 | 2025-02-27 | Pengeluaran |     50000 |      Top Up game |
|  4 | 2025-02-20 |   Pemasukan |    500000 | Menang DoorPrize |
+----+------------+-------------+-----------+-----------------+

========================================
|          MENU SORTING CATATAN        |
========================================
| 1. Urutkan berdasarkan Sumber (Desc) |
| 2. Urutkan berdasarkan Jumlah (Asc)  |
| 3. Urutkan berdasarkan Kategori (Asc)|
| 4. Kembali ke Menu Utama             |
========================================
Pilih menu (1-4): 3

Catatan berhasil diurutkan berdasarkan Kategori (Ascending)


================ Daftar Catatan ================
+----+------------+-------------+-----------+-----------------+
| No |   Tanggal  |  Kategori   |   Jumlah  |      Sumber     |
+----+------------+-------------+-----------+-----------------+
|  1 | 2025-04-20 |   Pemasukan |     20000 |          Ngojek |
|  2 | 2025-02-20 |   Pemasukan |    500000 | Menang DoorPrize |
|  3 | 2025-04-29 | Pengeluaran |     15000 |      Beli naspad |
|  4 | 2025-02-27 | Pengeluaran |     50000 |      Top Up game |
+----+------------+-------------+-----------+-----------------+
```

Gambar 4.8 Sorting jumlah pendapatan & pengeluaran secara ascending & sorting kategori secara ascending



Gambar 4.9 Kembali ke menu utama, logout, dan keluar dari program

5. Langkah-Langkah Git pada VSCode

1. Git add .

   Digunakan untuk menambahkan semua perubahan (termasuk file baru, modifikasi, dan penghapusan) dalam direktori kerja ke staging area, yang berarti file tersebut siap untuk di commit

```
PS D:\praktikum-apl> git add .
```

Gambar 5.1 git add .

2. Git commit -m “Selesai source code dan 90% Laporan”

   Digunakan untuk menyimpan perubahan yang ada di staging area ke repositori lokal, dengan menyertakan pesan commit yang menjelaskan perubahan tersebut.

```
PS D:\praktikum-apl> git commit -m "Selesai Source code dan 90% Laporan"
[master e4b628c] Selesai Source code dan 90% Laporan
 5 files changed, 609 insertions(+)
 create mode 100644 kelas/Pertemuan-6/ContohKasus.cpp
 create mode 100644 kelas/Pertemuan-6/ContohKasus.exe
 create mode 100644 post-test/post-test-6/2409106040-MuhammadRafliPernanda-PT-6.cpp
 create mode 100644 post-test/post-test-6/2409106040-MuhammadRafliPernanda-PT-6.exe
 create mode 100644 post-test/post-test-6/2409106040-MuhammadRafliPernanda-PT-6.pdf
```

Gambar 5.2 git commit -m

3. Git push

   Digunakan untuk mengirimkan perubahan file setelah di commit ke remote repository.

```
PS D:\praktikum-apl> git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 1.72 MiB | 601.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/RafliPernanda/praktikum-apl.git
   ce3e497..e4b628c  master -> master
```

Gambar 5.3 Git push