

# **LAPORAN PRAKTIKUM**

## **Modul 1**

### **“CODEBLOCKS IDE & PENGENALAN BAHASA C++ (BAGIAN 1)”**



**Disusun Oleh:**

**Alya Rabani - 2311104076**

**S1SE-07-B**

**Dosen :**

**Wahyu Andi Saputra, S.Pd., M.Eng**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY**

**PURWOKERTO**

**2024**

## 1. Tujuan

- Memperkenalkan bahasa pemrograman C++
- Mengetahui cara penginstalan aplikasi Codeblocks yang akan digunakan
- Memperkenalkan penulisan kode dalam bahasa C++
- Memahami penggunaan tipe data dan variabel
- Menggunakan operator-operator input/output dengan tepat
- Mengimplementasi dan memahami fungsi kondisi pada program

## 2. Landasan Teori

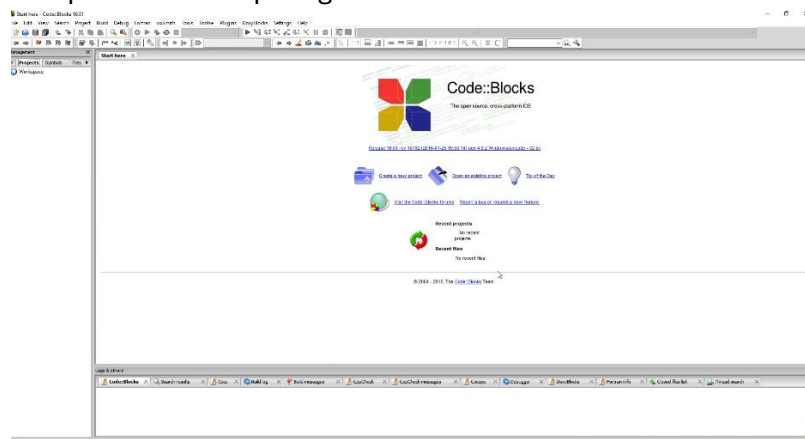
C++ adalah bahasa pemrograman yang dikembangkan sebagai peningkatan dari bahasa C untuk memasukkan paradigma pemrograman berorientasi objek. C++ merupakan bahasa yang imperatif dan dikompilasi, dirancang untuk pemrograman sistem dan aplikasi.

Pada praktikum struktur data ini, digunakan teks editor atau IDE yaitu Code::blocks. Codeblocks yakni lingkungan development terintegrasi untuk pemrograman bahasa C/C++ dan Fortran. Berbagai macam free Compiler tersedia di sana, mulai dari GNU GCC Compiler, Cygwin Compiler, Borland C++ Compiler, sampai dengan compiler untuk mikrokontroler seperti GNU Compiler for AVR, ARM, MSP430, PowerPC, Keil C51 Compiler, IAR 8051 compiler, IAR ARM compiler, dan masih banyak lagi. Codeblocks ini bersifat multi-platform atau cross-platform, artinya dapat diinstal di berbagai Sistem Operasi seperti Windows, Linux dan MacOS tanpa kehilangan fitur-fitur utamanya. Semuanya akan sama persis di setiap sistem operasi tempat instalannya.

### ❖ Instalasi Codeblocks IDE

Setelah mengetahui tools yang akan digunakan maka dilakukanlah penginstalan tools tersebut. Berikut cara penginstalan IDE codeblocks.

1. Instal terlebih dahulu aplikasi codeblocks dari link atau situs yang aman dan terpercaya. Setelah masuk akan diberikan beberapa cara untuk mendownload aplikasi tersebut, pilih download the binary release. Kemudian pilih pengaturan sesuai dengan operating sistem yang digunakan. Jika sudah pilih mingw-setup (e.g. codeblocks-20.03mingw-setup.exe) lalu tunggu hingga proses penginstalan selesai.
2. Jika semua proses penginstalan sudah selesai, saat membuka codebloks akan muncul tampilan utama seperti gambar berikut.

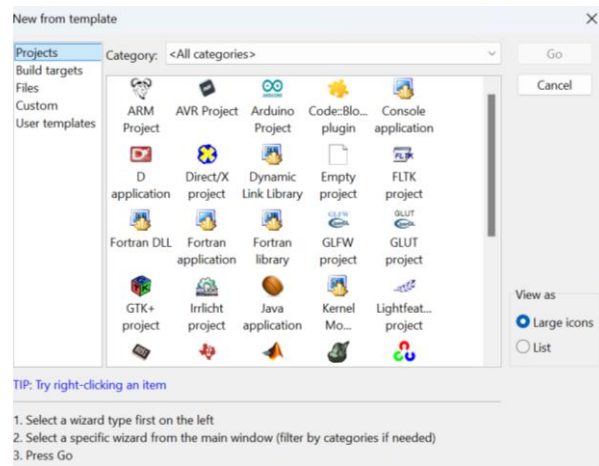


Gambar 1-1 Codeblocks IDE

❖ Penggunaan Codeblocks

Setelah penginstalan selesai maka langkah selanjutnya adalah bagaimana cara penggunaan dari Codeblock tersebut.

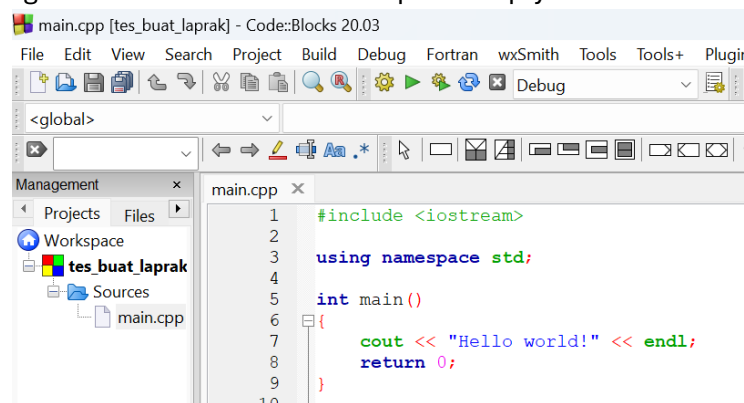
1. Untuk membuat dan menjalankan C++ di Codeblocks, pertama jika telah masuk di tampilan utama dari codeblocks maka pilih file > new > projects. Pada halaman project akan menampilkan beberapa macam pilihan menu seperti gambar berikut.



Gambar 1-2 membuat proyek baru

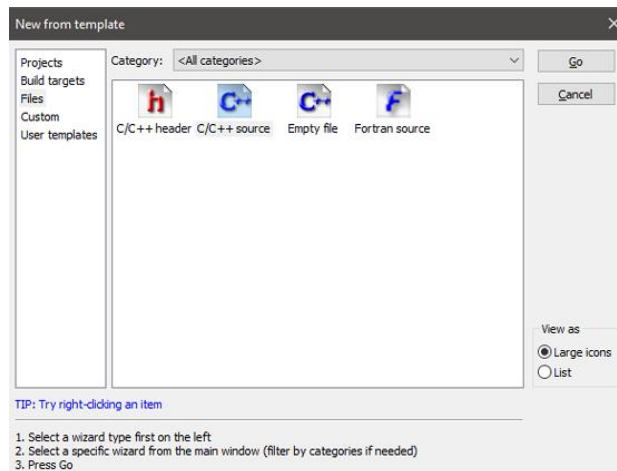
Lalu pilih console application > kemudian pilih bahasa pemrograman C++ > masukkan project title (nama proyek tidak boleh duplikasi) > lalu finish.

2. Jika sudah selesai maka akan terlihat sebuah empty file, disebelah kiri terdapat workplace, lalu ada project yang telah dibuat klik dua kali project tersebut > akan terdapat source klik lagi 2 kali source > akan ada main.cpp setelah mengklik 2 kali lagi akan muncul sintaks editor pada empty file tadi.



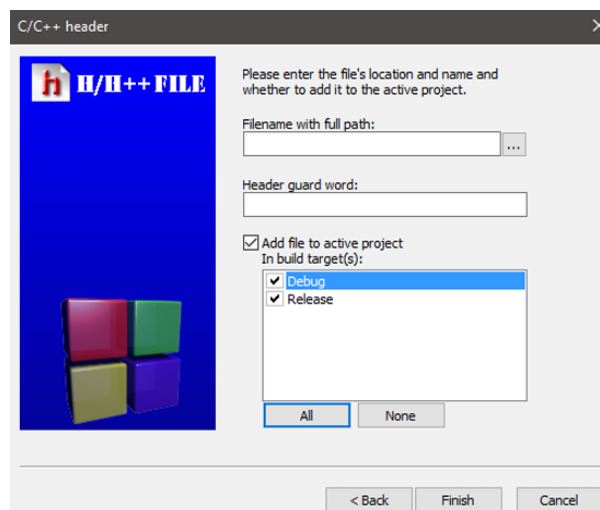
Gambar 1-3 penulisan sintaks

3. Untuk membuat kelas baru, ikuti langkah-langkah ini: Pertama, buka menu File dan pilih opsi New, lalu klik File. Selanjutnya, di panel sebelah kiri, cari dan pilih bagian Files. Pada panel di sisi kanan, carilah opsi C/C++ source. Terakhir, tekan tombol Go untuk melanjutkan proses pembuatan kelas baru.



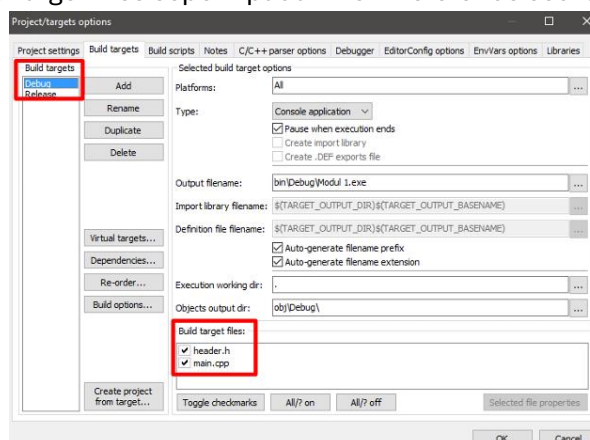
Gambar 1-4 buat class baru

Kemudian klik Next > Pilih bahasa pemrograman > Isi Filename with full path > Centang all in build target > Finish, seperti pada Error! Reference source not found..



Gambar 1-5 centang all in build target

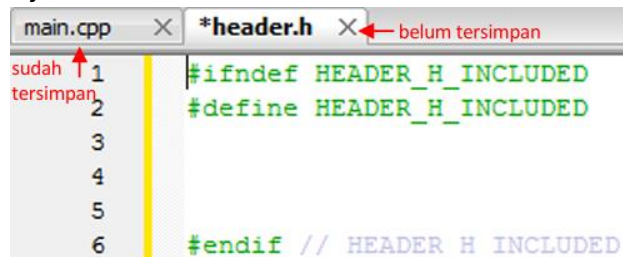
Jika anda lupa mencentang build target, dapat dilakukan setting manual dengan cara klik kanan pada project > properties > Build targets > Debug > Centang semua target files seperti pada Error! Reference source not found..



Gambar 1-6 target options

Setiap kelas perlu memiliki identitas unik berupa nama yang membedakannya dari kelas-kelas lainnya. Dalam penamaan kelas, gunakanlah kata benda

tunggal yang paling tepat mencerminkan abstraksinya. Konvensi penulisan nama kelas mengharuskan huruf awal setiap kata ditulis kapital, misalnya Jurusan dan NamaMahasiswa. Jangan lalai untuk menyimpan hasil kerja Anda secara berkala. Pintasan keyboard untuk menyimpan file tunggal adalah Ctrl+S, sementara Ctrl+Shift+S digunakan untuk menyimpan seluruh file sekaligus. Kelas yang belum disimpan akan ditandai dengan simbol bintang (\*) di sebelah kiri nama kelasnya.



Gambar 1-7 contoh file

4. Compile program, yaitu aksi untuk menjalankan program yang telah ditulis sehingga penulis dapat melihat output dari program.



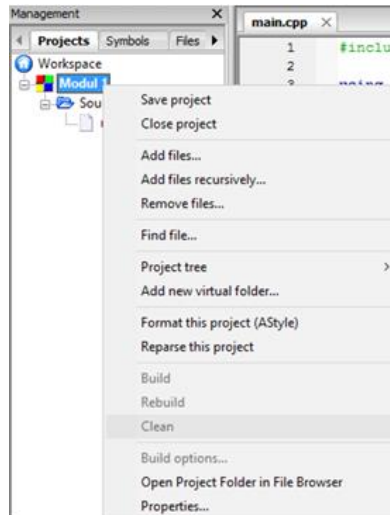
Gambar 1-8 compile program

Berikut adalah makna dan fungsi dari masing-masing opsi tersebut.

- Build, Opsi ini digunakan untuk mengompilasi kode sumber tanpa menjalankannya. Ini akan menghasilkan file eksekusi jika tidak ada kesalahan dalam kode. Build berguna untuk memeriksa kesalahan kompilasi.
- Run, Opsi ini menjalankan program yang telah dikompilasi sebelumnya. Jika ada perubahan pada kode, pastikan untuk melakukan build terlebih dahulu agar program yang dijalankan adalah versi terbaru.
- Build and Run, Opsi ini menggabungkan proses build dan run dalam satu langkah. Jika ada perubahan pada kode, Codeblocks akan mengompilasi ulang program dan kemudian langsung menjalankannya.
- Rebuild, Opsi ini mengompilasi ulang semua file sumber dalam proyek, meskipun tidak ada perubahan yang dilakukan. Ini berguna ketika ada masalah yang mungkin disebabkan oleh file yang tidak diperbarui.
- Abort, Opsi ini digunakan untuk menghentikan proses kompilasi atau eksekusi program yang sedang berlangsung. Ini bisa berguna jika proses berlangsung lebih lama dari yang diharapkan atau jika terdapat kesalahan yang tidak diinginkan.

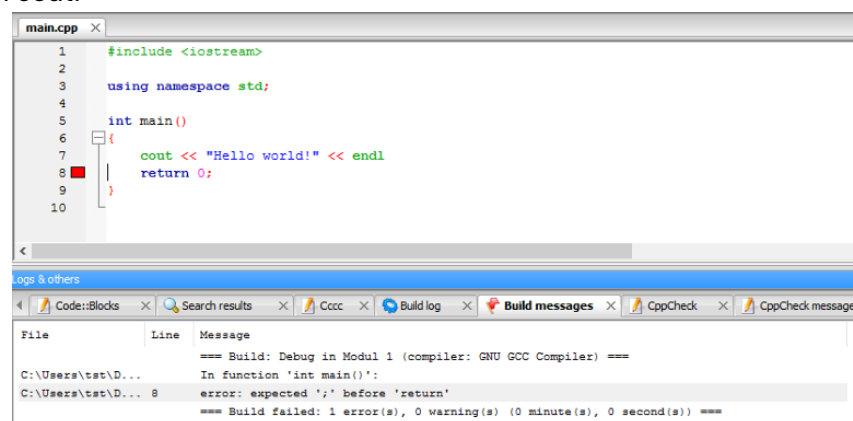
Setiap fungsi ini membantu dalam alur kerja pengembangan perangkat lunak, memungkinkan pengembang untuk mengelola dan mengeksekusi kode dengan lebih efisien.

5. Clean. Terkadang program yang kita buat tidak dapat di run, untuk itu perlu dilakukan clean project dengan cara klik kanan pada project kemudian klik Clean seperti pada Error! Reference source not found.. Setelah dilakukan clean, program akan dapat berjalan kembali.



Gambar 1-9 clean

6. Close dan Open Project. Untuk menutup project dilakukan dengan cara klik kanan pada project kemudian pilih close. Project yang di close tidak akan terhapus dan tetap ada pada directory. Sedangkan untuk membuka project yang telah di close dilakukan dengan cara memilih File > Open > Pilih File \*.cbp
7. Error Message akan muncul ketika terjadi kesalahan pada penulisan sintak. Contoh seperti pada Error! Reference source not found., terdapat error pada line 8. Pada error message diharapkan semicolon (;) sebelum return dan ternyata pada line 7 dapat dilihat bahwa penulisan sintak belum diakhiri untuk fungsi cout.



Gambar 1-10 error message

### 3. Guided

- Mendeklarasikan variabel dan mencetak variabel  
 Pada deklarasi variabel dalam C++ adalah proses memberi tahu compiler tentang nama variabel yang akan digunakan dalam program, serta jenis data yang akan disimpan di dalamnya. Menentukan jenis data yang dapat disimpan dalam variabel, seperti, int: untuk bilangan bulat (misalnya, 10, -5), float atau double: untuk bilangan desimal (misalnya, 3.14, -0.5), char: untuk karakter tunggal (misalnya, 'A', 'b'), dan string: untuk kumpulan karakter (misalnya, "Hello").

Dalam mencetak variable agar ditampilkan ke layer monitor dibutuhkan beberapa perintah seperti, cout Ini adalah perintah dalam C++ untuk menampilkan sesuatu ke layar. << Tanda panah ganda ini digunakan untuk memasukkan data ke dalam aliran output (dalam hal ini, layar). Kemudian ada teks yang akan ditampilkan secara langsung, tanpa perubahan. Lalu nama variabel yang berisi sesuatu yang ingin kita tampilkan. Dan endl yang merupakan perintah untuk membuat baris baru setelah output, sehingga tampilan menjadi lebih rapi. Contoh penggunaannya:

```

1 | #include <iostream>
2 | #include <conio.h>
3 |
4 | using namespace std;
5 |
6 | int main()
7 | {
8 |     int angka = 10;
9 |     float desimal = 10.5;
10 |    string kalimat = "alya";
11 |    double tinggi = 10.4;
12 |    char jenis_kelamin = 'P';
13 |    bool isSunny = true;
14 |
15 |    cout << "Angka: " << angka << endl;
16 |
17 |    cout << "Masukan angka: ";
18 |    cin >> angka;
19 |
20 |    cout << "Angka: " << angka << endl;
21 |
22 |    getch();
23 | }

```

Gambar 1-11 deklarasi dan mencetak variable

Output yang dihasilkan adalah:

```

C:\GUIDED\main.exe
Angka: 10
Masukan angka: 8
Angka: 8

Process returned 0 (0x0)   execution time : 7.632 s
Press any key to continue.

```

Gambar 1-12 output program

### ➤ Operator aritmatika

Operator aritmatika adalah simbol-simbol khusus yang digunakan untuk melakukan operasi perhitungan matematika dasar dalam bahasa pemrograman C++. Operasi ini meliputi penjumlahan, pengurangan, perkalian, pembagian, dan modulus (sisa hasil bagi). Berikut contoh penggunaan operator matematika menggunakan Bahasa pemrograman C++ pada codeblocks:

```

99 | #include <iostream>
100 | #include <conio.h>
101 |
102 | using namespace std;
103 |
104 | int main()
105 | {
106 |     // operator aritmatika
107 |     int angka1 = 5;
108 |     int angka2 = 10;
109 |
110 |     int hasil = angka1 % angka2;
111 |
112 |     cout << "Hasilnya adalah " << hasil << endl;
113 |
114 | }

```

Gambar 1-13 operator matematika

Gambar tersebut merupakan contoh penggunaan operator modulus, di mana int angka1 dan angka 2 merupakan variable yang menyebutkan nilai yang akan di bagi habis, kemudian int hasil merupakan pembagian dari angka1 dan angka2, kemudian untuk menampilkan hasil perhitungannya maka dipanggilah kode terakhir tersebut sehingga Ketika di jalankan ouput dari kode menjadi seperti berikut:

```
Hasilnya adalah 5
Process returned 0 (0x0)   execution time : 1.049 s
Press any key to continue.
```

Gambar 1-14 output dari program

➤ Operator perbandingan

Operator perbandingan dalam C++ digunakan untuk membandingkan dua nilai atau ekspresi. Hasil dari perbandingan ini adalah nilai boolean yaitu true (benar) atau false (salah). Operator perbandingan sering digunakan dalam struktur kontrol seperti pernyataan if, loop while, dan lainnya untuk mengontrol alur program berdasarkan kondisi tertentu. Berikut contoh penggunaannya:

```
116 #include <iostream>
117 #include <conio.h>
118
119 using namespace std;
120
121 int main()
122 {
123     //operator perbandingan
124     int angka1 = 0;
125     int angka2 = 1;
126
127     bool hasil = (angka1 != angka2);
128
129     cout << "hasilnya adalah " << hasil << endl;
130 }
```

Gambar 1-15 operator perbandingan

Gambar tersebut merupakan operator perbandingan tidak sama dengan, diketahui variable angka1 dan angka2 jika dibandingkan akan tidak sama dengan sehingga outputnya akan menjadi seperti berikut:

```
hasilnya adalah 1
Process returned 0 (0x0)   execution time : 0.815 s
Press any key to continue.
```

Gambar 1-16 output dari program

➤ Operator logika

Operator logika dalam C++ digunakan untuk menggabungkan atau memodifikasi ekspresi boolean (yang menghasilkan true atau false). Operator ini sering digunakan dalam struktur kontrol seperti pernyataan if, loop while, dan lainnya untuk membuat keputusan berdasarkan beberapa kondisi. Contoh penggunaannya:



```

132 #include <iostream>
133 #include <conio.h>
134
135 using namespace std;
136
137 int main()
138 {
139     //operator logika
140     bool kondisi1 = true;
141     bool kondisi2 = false;
142
143     bool hasil = (kondisi1 && kondisi2);
144     cout << "Hasilnya adalah " << boolalpha << hasil << endl;
145
146     /*bool kondisi1 = false;
147     bool hasil = !kondisi1;
148     cout << "Hasilnya adalah " << boolalpha << hasil << endl;*/
149 }

```

Gambar 1-17 operator logika

Pada gambar tersebut dibuat 2 kondisi true dan false, pada hasilnya digunakan operator logika AND(&) sehingga jika program dijalankan maka hasilnya akan menjadi seperti gambar berikut:

```

Hasilnya adalah false

Process returned 0 (0x0)   execution time : 0.975 s
Press any key to continue.

```

Gambar 1-18 output dari program

#### ➤ Percabangan

Dalam C++, percabangan digunakan untuk mengontrol alur eksekusi program sehingga dapat menangani berbagai situasi yang mungkin terjadi saat program dijalankan. Ada berbagai macam perulangan yang digunakan seperti if Statement, if-else Statement, else-if Ladder, Percabangan Bertingkat (Nested if-else), switch-case Statement, Operator Ternary (? :)

```

151 #include <iostream>
152 #include <conio.h>
153
154 using namespace std;
155
156 int main()
157 {
158     //percabangan
159     string kata;
160     cout << "Masukan kata = Ohayoo" << endl;
161     cin >> kata;
162
163     if (kata == "Ohayoo") {
164         cout << "Kata sesuai" << endl;
165     } else {
166         cout << "Kata tidak sesuai" << endl;
167     }

```

Gambar 1-19 percabangan

Pada gambar tersebut merupakan if-else statement yang memungkinkan eksekusi salah satu dari dua blok kode berdasarkan kondisi yang diberikan. Sehingga jika dijalankan pengguna diminta untuk menuliskan kalimat yang diperitahkan jika sama maka hasilnya adalah kata sesuai dan jika salah kata tidak sesuai.

```

Masukan kata = Ohayoo
Ohayoo
Kata sesuai

Process returned 0 (0x0)   execution time : 7.127 s
Press any key to continue.

```

Gambar 1-20 output dari program

#### ➤ Perulangan

Dalam C++, terdapat beberapa jenis perulangan yang dapat digunakan sesuai dengan kebutuhan program Anda. Dengan memahami berbagai jenis perulangan serta cara penggunaannya, Anda dapat menulis program yang lebih

efisien dan fleksibel. Ada berbagai jenis-jenis perulangan dalam C++ seperti, for loop, while loop, do-while loop, perulangan bertingkat (nested loops), break dan continue statement.

```
194 #include <iostream>
195 #include <conio.h>
196
197 using namespace std;
198
199 int main()
200 {
201     //perulangan
202
203     for(int i = 0; i<5; i++){
204         cout << "hello world" << endl;
205     }
206
207     for(int i=0; i<5; i++){
208         cout << "hello world" << endl;
209     }
210 }
```

Gambar 1-21 Perulangan

Dari gambar tersebut merupakan perulangan for loop yang digunakan ketika jumlah iterasi (pengulangan) sudah diketahui atau dapat diprediksi sebelumnya. Struktur for loop terdiri dari tiga bagian utama: inisialisasi, kondisi, dan iterasi. Sehingga output dari program tersebut menjadi:

```
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world

Process returned 0 (0x0)   execution time : 5.027 s
Press any key to continue.
```

Gambar 1-22 output dari program

## 4. Unguided

1. Pertama program meminta pengguna untuk memasukkan dua bilangan bertipe float, lalu program melakukan empat operasi aritmatika: penjumlahan, pengurangan, perkalian, dan pembagian. Untuk pembagian, program melakukan pengecekan apakah bilangan kedua bernilai nol untuk menghindari kesalahan pembagian dengan nol. Maka, output dari program mencetak hasil dari setiap operasi aritmatika yang dilakukan.

Kode program:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      float bilangan1, bilangan2;
8
9      cout << "Masukkan bilangan pertama: ";
10     cin >> bilangan1;
11     cout << "Masukkan bilangan kedua: ";
12     cin >> bilangan2;
13
14     float penjumlahan = bilangan1 + bilangan2;
15     float pengurangan = bilangan1 - bilangan2;
16     float perkalian = bilangan1 * bilangan2;
17
18     if (bilangan2 != 0) {
19         float pembagian = bilangan1 / bilangan2;
20         cout << "Hasil Pembagian: " << pembagian << endl;
21     } else {
22         cout << "Pembagian dengan nol tidak dapat dilakukan!" << endl;
23     }
24
25     cout << "Hasil Penjumlahan: " << penjumlahan << endl;
26     cout << "Hasil Pengurangan: " << pengurangan << endl;
27     cout << "Hasil Perkalian: " << perkalian << endl;
28
29     return 0;
30 }
```

Output dari program:

```
C:\Unguided1\bin\Debug\Unguided1.exe
Masukkan bilangan pertama: 28.9
Masukkan bilangan kedua: 0.4
Hasil Pembagian: 72.25
Hasil Penjumlahan: 29.3
Hasil Pengurangan: 28.5
Hasil Perkalian: 11.56

Process returned 0 (0x0)   execution time : 10.869 s
Press any key to continue.
```

2. Pada program tersebut pengguna diminta memasukkan angka. Angka tersebut dimasukkan kedalam bentuk array. Kedua array ini menyimpan kata-kata untuk satuan dan puluhan. Index array sesuai dengan nilai angka (misalnya, satuan[3] adalah "tiga"). Angka yang dimasukkan akan dikonversi menjadi kata menggunakan fungsi. Pada program utama meminta pengguna untuk memasukkan angka, kemudian memanggil fungsi Kembali, dan menampilkan hasil konversi.

Kode program:

```

1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 string satuan[] = {"", "satu", "dua", "tiga", "empat", "lima", "enam", "tujuh", "delapan", "sembilan"};
7 string puluhan[] = {"", "", "dua puluh", "tiga puluh", "empat puluh", "lima puluh", "enam puluh", "tujuh puluh", "delapan puluh", "sembilan puluh"};
8
9 string convertNumberToWords(int number) {
10     if (number < 10) {
11         return satuan[number];
12     } else if (number < 20) {
13         return "belas";
14     } else {
15         return puluhan[number / 10] + " " + satuan[number % 10];
16     }
17 }
18
19 int main() {
20     int angka;
21
22     cout << "Masukkan angka (0-100): ";
23     cin >> angka;
24
25     if (angka >= 0 && angka <= 100) {
26         string kata = convertNumberToWords(angka);
27         cout << angka << ": " << kata << endl;
28     } else {
29         cout << "Angka harus antara 0 dan 100." << endl;
30     }
31     return 0;
32 }
33
34

```

Output dari program:

```

C:\Unguided2\bin\Debug\Un... X + v
Masukkan angka (0-100): 79
79: tujuh puluh sembilan

Process returned 0 (0x0)   execution time : 8.199 s
Press any key to continue.

```

3. Pertama program akan meminta pengguna untuk memasukkan jumlah baris yang diinginkan. Lalu, dibuat looping luar untuk mengontrol jumlah baris yang akan dicetak dan ada juga looping dalam pertama mencetak angka dari i hingga 1, membentuk pola angka menurun pada setiap baris. Kemudian, dibuat cetak tanda bintang di tengah untuk memisahkan angka menurun dan menaik. Dibuat lagi looping dalam kedua mencetak angka dari 1 hingga i, membentuk pola angka menaik (mirror) pada setiap baris. Dan Looping tambahan mencetak tanda bintang sebanyak n kali untuk baris terakhir.

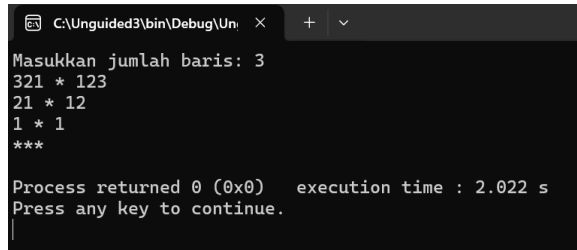
Kode program:

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int n;
8
9     cout << "Masukkan jumlah baris: ";
10    cin >> n;
11
12    for (int i = n; i >= 1; i--) {
13
14        for (int j = i; j >= 1; j--) {
15            cout << j;
16        }
17
18        cout << " * ";
19
20        for (int j = 1; j <= i; j++) {
21            cout << j;
22        }
23
24        cout << endl;
25    }
26
27    for (int i = 1; i <= n; i++) {
28        cout << " * ";
29    }
30    cout << endl;
31
32    return 0;
33 }

```

Output dari program:

A screenshot of a debugger window titled 'C:\Unguided3\bin\Debug\Un...'. The window shows the output of a C++ program. The first line is 'Masukkan jumlah baris: 3'. The next three lines are '321 \* 123', '21 \* 12', and '1 \* 1'. This is followed by three asterisks '\*\*\*'. The final line of output is 'Process returned 0 (0x0) execution time : 2.022 s' and 'Press any key to continue.'.

```
C:\Unguided3\bin\Debug\Un...
Masukkan jumlah baris: 3
321 * 123
21 * 12
1 * 1
***
Process returned 0 (0x0) execution time : 2.022 s
Press any key to continue.
```

## 5. Kesimpulan

Laporan ini menunjukkan bahwa mahasiswa berhasil memahami dasar-dasar pemrograman C++ serta cara menggunakan Codeblocks IDE. Mahasiswa juga mendapatkan wawasan penting tentang konsep fundamental pemrograman seperti penulisan kode, tipe data, variabel, dan operator dasar. Pemahaman ini menjadi fondasi yang sangat berharga untuk pengembangan keterampilan di tahap berikutnya, terutama dalam menyusun program yang lebih kompleks. Selain itu, laporan ini memberikan panduan praktis dalam menangani masalah yang sering muncul dalam pemrograman, seperti kesalahan kode (error) yang memerlukan debugging dan compile. Mahasiswa juga diperkenalkan pada penggunaan struktur kontrol alur, seperti perulangan dan percabangan, yang membantu mengelola logika program dengan lebih baik. Dengan melakukan praktik langsung, mahasiswa tidak hanya mempelajari teori, tetapi juga merasakan pengalaman nyata dalam menyelesaikan masalah pemrograman. Meskipun terdapat tantangan seperti kesalahan sintaks, laporan ini dengan jelas menunjukkan bagaimana masalah tersebut dapat diatasi, yang pada akhirnya memperkaya pemahaman dan keterampilan pemrograman mereka.